Thèse de doctorat de l'Université de Paris
Ecole Doctorale 386 - Sciences Mathématiques de Paris Centre
Institut de Recherche en Informatique Fondamentale (IRIF)

Logical foundations of a modelling assistant for molecular biology

Fondements logiques d'un assistant à la modélisation en biologie moléculaire

par Adrien HUSSON

Thèse de doctorat d'Informatique
Dirigée par Jean KRIVINE

Thèse soutenue publiquement le 16 décembre 2019 devant le jury composé de :

Jean Krivine	Directeur de thèse	Chargé de recherche	Université de Paris
Nicolas Peltier	Rapporteur	Chargé de recherche	Université de Grenoble
François Fages	Rapporteur	Directeur de recherche	INRIA Saclay
Ralf Treinen	Président du jury	Professeur	Université de Paris
Anne Siegel	Examinatrice	Directrice de recherche	Université de Rennes
Walter Fontana	Examinateur	Professeur	Harvard University
Hubert Comon	Examinateur	Professeur	ENS Paris-Saclay

Résumé

Résumé Cette thèse concerne la "représentation exécutable du savoir"dans le domaine de la biologie moléculaire. Elle introduit les fondements d'un cadre logique appelé iota, dont le but est de décrire et rassembler des faits au sujet d'interactions entre protéines tout en offrant au modeleur la possibilité de "compiler" un fragment raisonnable de la logique vers un ensemble fini de règles de réécriture.

On définit une logique FO[↓] qui décrit des transitions d'états cellulaires. Un état représente le contenu d'une cellule : les éléments du domaine sont des parties de protéines et les relations sont des liaisons entre protéines. L'opérateur logique unaire ↓ sélectionne les transitions où un ensemble minimal de changements a lieu. Les formules qui parlent de transitions dénotent aussi des exécutions, c'est-à-dire des séquences finies ou infinies de transitions. Chaque formule de transition est de plus associée à un ensemble de règles de réécritures équipé d'une sémantique opérationnelle. On introduit deux système déductifs qui permettent de "typer" les formules. On montre que si une formule est typable dans le 1er système, alors l'exécution des règles de réécriture qui lui sont associées produit exactement les exécutions dénotées par la formule ; et que si elle est typable dans le 2nd système, alors son système de règles associé est fini. On introduit une grammaire qui produit des formules typables dans les deux systèmes (à équivalence logique près). Enfin, on étudie la décidabilité et l'expressivité de fragments de FO[↓]. On montre en particulier que les formules typables dans le second système sont définissables dans un petit fragment de FO, ce qui implique que l'opérateur ↓ peut alors être éliminé.

Mots-clefs Raisonnement non-monotone, representation des connaissances, modélisation basée sur les règles de réécriture, biologie moléculaire, logique du 2nd ordre, circonscription

Abstract

This thesis addresses the issue of "Executable Knowledge Representation" in the context of molecular biology. We introduce the foundation of a logical framework, termed iota, whose aim is to facilitate knowledge collation of molecular interactions at the level of proteins and at the same time allows the modeler to "compile" a reasonable fragment of the logic into a finite set of executable graph rewriting rules.

We define a logic FO[↓] over cell state transitions. States represent cell contents; domain elements are protein parts and relations are protein-protein bindings. The unary logical operator ↓ selects transitions where as little as possible happens. Formulas over transitions also denote runs, which are finite or infinite sequences of transitions. Every transition formula is moreover associated to a set of rewriting rules equipped with an operational semantics. We introduce two deductive systems that act as "typing" for formulas. We show that if a formula is typable in the first system then the execution of its associated rule set produces exactly the runs denoted by the formula, and that if it is typable in the second system then its associated rule set is finite. We introduce a grammar that produces formulas typable in both systems, up to logical equivalence. Finally we study decidability and definability properties of fragments of FO[↓]. In particular, we show that formulas typable in the second system are in a tight fragment of FO, which implies that the operator ↓ can then be eliminated.

Keywords non-monotonic reasoning, knowledge representation, rule-based modelling, molecular biology, second-order logic, circumscription

Contents

1. Introduction

1.1. Biological modeling

Biological systems involve many components that interact non-linearly. These components can be molecules, cells, genes, etc. The level of complexity of many of these systems precludes a full mental grasp over all their possible behaviors.

For the past several decades, the field of systems biology has developed mathematical models that have become an important tool for biological research. These models explicitly describe the behaviors and interactions of biological systems; they can then be interpreted by a computer for testing and prediction purposes.

We start with a very simple example designed to motivate biological modelling in general.

A boolean network example We introduce biological modeling with a simple type of model: a boolean network. A boolean network can represent how proteins influence one another through their activity. The toy model in figure 1.1 shows three protein types: A, B and C, as well as their influence on one another's activity. The activity of a protein is summarised as "active" and "inactive". If a protein or a pair of proteins is active, it can turn on or turn off the activity of another protein. The green arrows in figure 1.1 represent activation (positive influence), and the red, flat arrows represent inhibition (negative influence). A double arrow requires both sources to (de)activate. For instance, A alone has a negative influence on B, but A and B together have a positive influence on C. Double arrows have priority over single arrows.

At any moment in time, each of A, B and C can be active or inactive (but we assume that it is not possible for all 3 to be inactive at the same time). That can be summarised by 3 letters ABC. If a letter is green, the corresponding protein is active. Otherwise it is inative. For instance, ABC means that only A and C are active. Each possible triple of green or black letters is called a *state*.

Using the individual and pair influences on figure 1.1, we can generate a *trace* from any starting state. That means starting with any activation values for A, B and C, and then iteratively

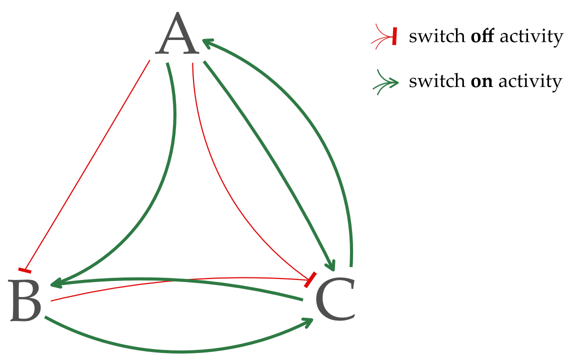


Figure 1.1.: Activity influences between A,B and C. A joint influence has priority.

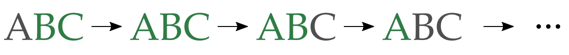


Figure 1.2.: Generating traces

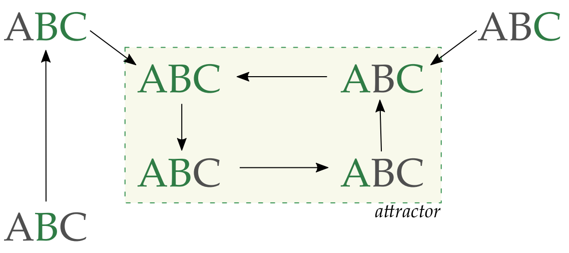


Figure 1.3.: The resulting network. A green letter is active.

updating those values according to the influence map of figure 1.1.

For instance, figure 1.2 shows a sample trace. The trace starts in state ABC, which means that A is inactive and both B and C are active. Because C has a positive influence on A (cf. the green arrow from C to A in the influence map of figure 1.1), that state can evolve to ABC. Then, because A and B together have a negative influence on C (cf. the double, flat red arrow from (A, B) to C in the influence map), the state can evolve to ABC, and so on.

If we generate many such traces, we will find out that the states ABC, ABC and ABC are not seen very often. In fact, unless we start in one of these states, we never see them at all!

To better understand the situation, we draw a map of all the states and of how traces can move from each one to the others. The resulting diagram is shown in figure 1.3. Each state has A,B,C active or inactive, and arrows are possible moves by the trace-generation process. For instance, there is a (leftward) arrow from ABC to ABC because A and C together have a positive influence on B.

If we look at the diagram carefully, we notice an attractor region (yellow rectangle). On any trace, the system will go to and stay among the attractor's states: since there is no arrow coming out of the yellow region, it is impossible to leave it.

That last observation illustrate the essence of biological modeling: going from a set of individual facts (the joint influences of fig. 1.1) to an insight on the global dynamics (the attractor region of fig. 1.3) through the manipulation and analysis of a mathematical object.

In this thesis, we will be concerned with the earliest part of the modelling process: how to describe the individual facts and collate them together. We also won't be working on boolean networks but on different formal objects, which follow the paradigm of *rule-based modelling* [Boutillier et al., 2018, Chylek et al., 2014, Danos et al., 2007].

1.2. Rule-based models

The purpose of the previous section was to show the biological modelling process, from observations to a formal model to analysis of the model and what that may imply about the real world. We now introduce another biological modelling formalism: *rule-based models*. They will be the starting point of the objects we construct in this thesis.

Boolean networks are typically about modelling gene expression. They are not mechanistic, in the sense that they model statistical interactions at a rather large scale. There is an important, more mechanistic level of abstraction: protein-protein interactions, as described by molecular biology.

To illustrate the type of knowledge we are aiming at, here is a typical sentence from a molecular biology paper:

> *"The activation of Raf-1 by activated Src requires phosphorylation of Raf-1 on Y340 and/or Y341 [...]. Tyrosine phosphorylation and activation of Raf-1 have been shown to be co-incident. However, others have been unable to detect phosphotyrosine in active Raf-1.".* [Mason et al., 1999]

At this level of abstraction, proteins are considered as chains of amino acid residues such as Y340 and Y341, which are identified by their type (Y for tYrosine) and their position in the chain (resp. 340 and 341). Proteins have names, here Raf-1 and Src, and are usually divided into domains or regions that are covering sub-sequences of amino acids. Domains may also be given a name. For instance, Raf-1 has a "Zinc finger" domain in the 137-183 region.[1]

Importantly, *static* names of proteins, domains and residues can be completed with *dynamic* attributes. Here, "phosphorylation" denotes the attachment of a phosphate group to a protein residue, which tends to modify the protein structure. One then talks about a phosphorylated protein, a phosphorylated domain or, as in the example above, a phosphorylated residue. Other dynamic attributes such as "active" are commonplace in molecular biology.

Underlying the snippet of biological literature given above is the notion of protein interactions: "the activation of Raf-1" by "activated Src" indicates that Raf-1 and active Src can bind to each other so that phosphorylation of Raf-1 by Src may occur. Stable binding of proteins requires complementary domains that stick together with various affinities. The binding state of a protein (or a region) is therefore also a dynamic, relational property.

Over time, the field of molecular biology accumulates data and mechanisms suspected to play key roles in the cellular ecosystem. The activity of discovery currently outpaces human abilities to follow and collate new mechanisms [Lazebnik, 2002]. For instance, p53 is a protein family relevant to cell apoptosis and cancer formation. In 2018 alone, it was mentioned in the title or abstract of about 4700 papers (PubMed). The need for principled representations of biological systems has led to the development of formal methods for biology [Fontana and Buss, 1996].

The field of *rule-based modelling* is well-suited to formally modelling mechanistic biological interactions at that lower level of abstraction. Protein-protein interactions are directly described, and gene expression is a global observation on the model's behavior. Rule-based models attempt to collate a large number of known interactions in a single structure: a list of rewrite rules.

Mixtures A rule-based model is a list of *rules* that describe allowed interactions between proteins – think of chemical reactions, but with proteins and protein parts instead of individual atoms. The interactions take place in a *mixture*, a graph which represents the contents of a cell or of a test tube.

Figure 1.4 presents a possible mixture: it contains several *agents* of two types: K for Kinase, E for general enzyme (a kinase is a particular kind of enzyme which helps bind phosphate groups to other proteins). Each agent represents a protein. Some agents have been given names so they can be individually identified: there is an agent a (of type K) and an agent b (of type E). Some agents have *sites* (s, t): each site represents the ability of its protein to interact with another protein. Sites can change in two ways :

[1]uniprot.org/uniprot/P09560

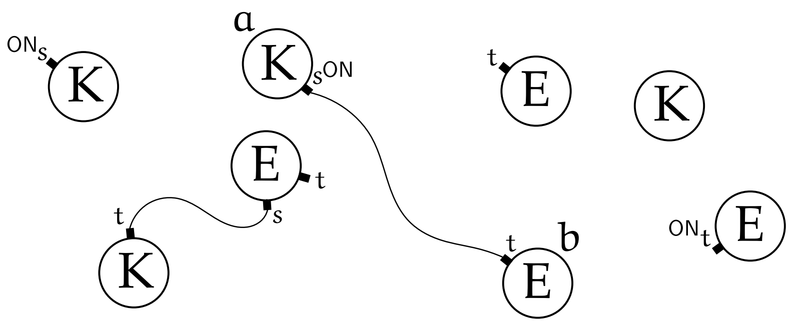


Figure 1.4.: A mixture

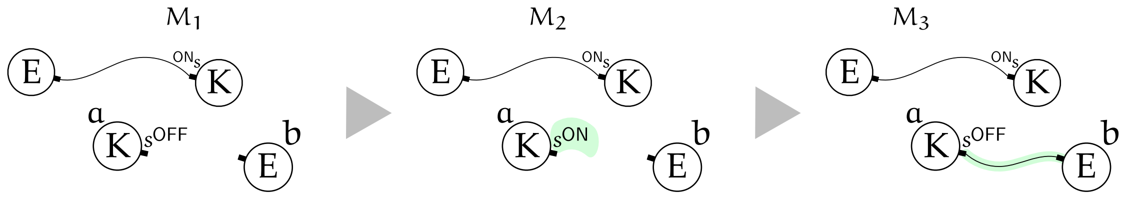


Figure 1.5.: A sequence of mixtures, changes highlighted in green

- The *label* of a site may change. In figure 1.4, some sites are ON, others not. Labels represent any dynamic aspect of a protein: its state of phosphorylation (that is, whether a phosphate group is attached to part of the protein), its folding conformation, etc.
- The *linkage* of a site may change. If two proteins are linked, they are interacting. As with labels, linkage is dynamic: it may change over time. In the mixture presented in figure 1.4, a and b are linked.

In figure 1.5, we see the sequence of mixtures M_1, M_2 and M_3. They can be seen as three snapshots of a mixture as it evolves in time:

- in M_1, agent a's site s (written a.s) is OFF. a.s and b.s are not linked;
- in M_2, a.s is ON. a.s and b.s are stil not linked.
- in M_3, b.s and a.s are linked.

Rules Driving the evolution of a mixture are the *rules* mentioned earlier. A rule allows a possible change. In figure 1.6, we see the rule[2] R_1. It allows the site s of any Kinase to spontaneously turn ON when it is not *linked* to anyone.

If an *instance* of a rule is found in a mixture, that mixture can evolve.

Figure 1.8 shows how to find instances of rules: A *match* for R_1 is found by identifying a part of M_1 "similar enough"[3] to the left-hand side of R_1 (cf. red arrow, figure 1.8). Then, the changes

[2]In figures, rules are represented with a dashed central triangle in the center, to differentiate from the mere succession of mixtures separated by a grey triangle.

[3]A match is a particular kind of morphism.

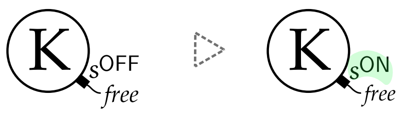


Figure 1.6.: The rule R_1

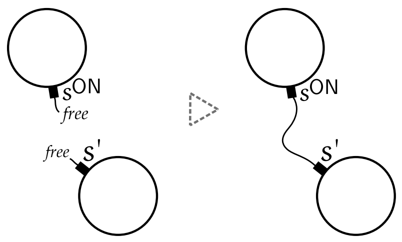


Figure 1.7.: The rule R_2

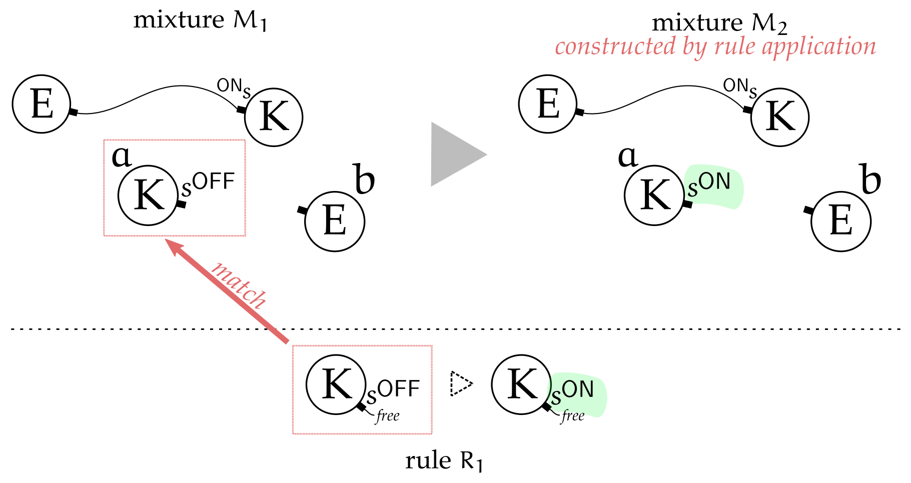


Figure 1.8.: A match from R_1 into M_1

in R_1 (i.e. going from OFF to ON) are applied to M_1 along the match arrow. This constructs the mixture M_2 (top right).

In R_1, notice the word *free* attached to s: it means that for the change to occur, the match image of s in M_1 should not be linked to anyone. The word *free* only appears in rules, not in mixtures, because it is not a property of the graph itself. Instead it is a *application condition* restricting where the rule can be applied.

Figure 1.7 shows the rule R_2, where any two agents can become linked as long as one has an ON site s and that both s sites are free. Following the example of figure 1.8, consider how the rule R_2 could be applied to M_2 in order to construct M_3.

Given a set of rules and an initial mixture one can generate a *trace* by repeating the above operation and at every step nondeterministically choosing one match among the possible ones. Figure 1.5 is actually an example such a sequence, starting from mixture M_1. It uses the rules R_1 and R_2 shown in figures 1.6 and 1.7.

Problems with rule-based models Rule-based models suffer from two problems that arise as modelling scales up. The first is that models become themselves too large understand. In particular if multiple groups collaborate, it becomes harder to write rules consistent with every other observation, encoding choice and biological invariant of the model. Since model building rests on current biological data, it should be possible to query the existing system; to maintain a coherent model, it is useful to answer queries such as "Is this rule subsumed by a preexisting

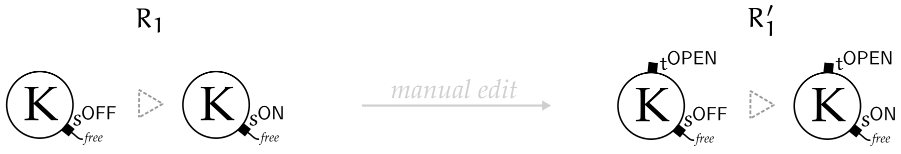


Figure 1.9.: R_1 becomes R_1'

one?" or "Is that biological invariant ensured by the current model?". In the next paragraph we introduce a logic-based description of biological knowledge. In such a framework, natural language queries of that type can be encoded as boolean or n-ary logical queries.

A second, related issue is that knowledge should be added *incrementally* as much as possible. Under the rule-based paradigm, it is hard to add new knowledge to an existing set of rules. Sometimes that knowledge is about a new & previously unknown reaction. Then it suffices to create a new rule. But sometimes incorporating the new knowledge means *modifying* existing rules. Take the rule R_1 in figure 1.6, which lets a Kinase go from OFF to ON. Suppose that the following information comes in: *A Kinase turns ON only when the label of its t site is OPEN.* We would like that information to exist as a discrete piece of knowledge. But the only way to incorporate it is to edit every single rule where a kinase goes from OFF to ON, as in figure 1.9, where R_1 is edited into R_1'.

Knowledge representation When one wishes to formally represent information, turning to logic is usually a sensible move. Here is a logic formula; we call if F_0. The x in parenthesis means that it depends on an argument named x. The $:=$ symbol means "is defined to be", and $\wedge$ means "and":

$$F_0(x) := \mathsf{OFF}(x.s) \wedge \mathsf{ON}^\star(x.s)$$

$F_0(x)$ means : *The s site of agent x turns ON.*

- $\mathsf{OFF}(x.s)$ means that the site s of x is currently OFF.
- $\mathsf{ON}^\star(x.s)$ means that in the *next* mixture, the site s of x will be ON.

Note the presence of a star after the word ON. The star is for things that are susceptible to change (labels and links). No star means we are talking about the *current* mixture, while a star means we are talking about the *next* mixture.

Compare the formula F_0 to the rule R_1 in figure 1.6. Informally, note how they are partially similar: in both cases, some agent's site s goes form OFF to ON. For the rule R_1 the switch is represented by a pair of graph. For the formula F_0, the switch is encoded by two statements taken in conjunction.

We can refine F_0 into another formula, F_1, so that it corresponds more closely to our intuitive understanding of the rule R_1:

$$F_1 := F_0 \wedge K(x) \wedge \mathit{free}(x.s)$$

- $K(x)$ means "x is a Kinase". Names such as Kinase and Enzyme cannot change, so there is no need to also write $K^\star(x)$.
- $\mathit{free}(x.s)$ is another formula. It specifies that the site s of x should not be linked to anyone in the *current* mixture (there exists a formula $\mathit{free}^\star$ for the *next* mixture).

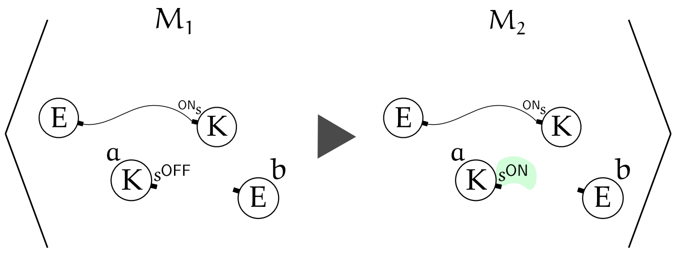


Figure 1.10.: The transition $\langle M_1, M_2 \rangle$

The meaning of a formula depends on how we *interpret* its variables. $F_0(a)$ means that a is turning ON. $F_0(b)$ means that b is turning ON.

With all its variables interpreted, a formula may or may not be *satisfied* by a pair of mixtures. A pair of mixtures can be represented by a *transition*. One mixture is the *current* one, while the other is the *next*. For instance, $\langle M_1, M_2 \rangle$ is the transition where M_1 is the current mixture, and M_2 is the next one (see figure 1.10)[4]. In $\langle M_1, M_2 \rangle$, $F_1(a)$ is satisfied because a is a Kinase, $a.s$ does turn ON between M_1 and M_2, and s is not linked to anyone. $F_1(b)$ is not satisfied because b does not even have a site s.

One advantage of using logic formulas is that it is easier to incorporate new knowledge. Recall the scenario where a new piece of information comes in: *A Kinase's* s *site may only turn ON when its* t *site is OPEN*. Let us write a formula G that says exactly this:

$$G := \forall x.\ (K(x) \wedge \mathsf{OFF}(x.s) \wedge \mathsf{ON}^\star(x.s)) \to \mathsf{OPEN}(x.t)$$

- The universal quantifier $\forall x.$ means that what follows it should be true *for all* interpretations of x, and the arrow $\to$ means "implies".
- As earlier, $K(x)$ means that x has the static name K, OFF is about the current state, and $\mathsf{ON}^\star$ is about the next mixture.
- The formula requires all Kinases in the transition with an s site that turns ON to also have a t site OPEN in the current mixture.

In the previous section, the only way to incorporate that new knowledge was to edit R_1 into R_1', as in figure 1.9. Now G is an independent unit of knowledge, and it suffices to write $F_1 \wedge G$.

Traces As we saw earlier, rules can be *executed*. So far there is no notion of execution associated with formulas. Formulas have semantics and we may reason with them but we cannot generate traces with them.

Consider what it would mean for a trace to satisfy (or not) a formula. We wrote the formula F_1 to correspond to the rule R_1. We will also write $F_2(x, y)$ to correspond to the rule R_2:

$$F_2(x, y) := \textit{free}(x.s) \wedge \textit{free}(y.t) \wedge \mathsf{ON}(y.s) \wedge \mathsf{Link}^\star(x.s, y.t)$$

- The formula F_2 depends on two variables, x and y.
- *free*$(x.s)$ stands for "nothing is connected to $x.s$". Same for $y.t$.

[4]In figures we use a dark grey triangle for transitions. It helps distinguishing them from rules (dashed triangle) and sequences of mixtures (light grey triangle).

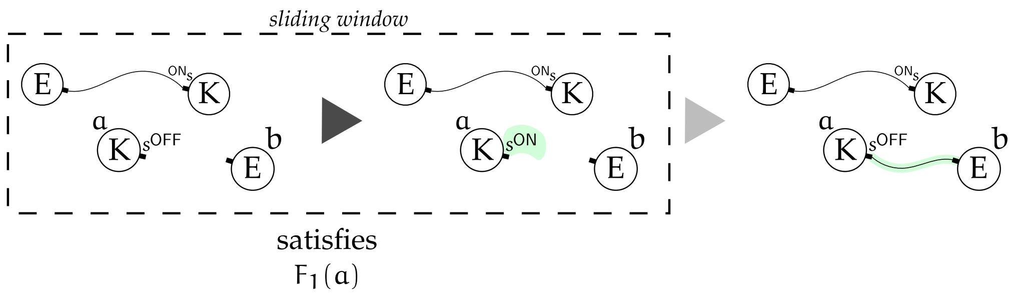


Figure 1.11.: The transition from M_1 to M_2 satisfies $F_1(a)$, and therefore F

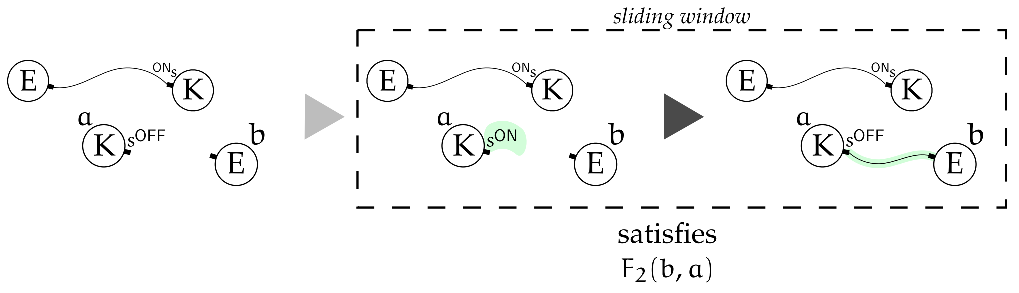


Figure 1.12.: The transition from M_2 to M_3 satisfies $F_2(b, a)$, and therefore F

- $\mathsf{Link}^\star(x.s, y.t)$ stands for "$x.s$ and $y.t$ are connected in the *next* mixture".

The formula F allows both F_1 and F_2 ($\vee$ means "or"):

$$F(x, y) := F_1(x) \vee F_2(x, y)$$

Consider again the sequence M_1, M_2, M_3 shown in figure 1.5 on page 4. The sequence M_1, M_2, M_3 is correct for the rules R_1 and R_2 because each step corresponds to the application of a rule among $\{R_1, R_2\}$. We will say that M_1, M_2, M_3 is correct for F if every successive pair of mixtures in it satisfies F.

This is illustrated in figures 1.11 and 1.12. Picture a "sliding window" wide enough for 2 mixtures, sliding over the sequence. The pair of mixtures visible under the window is a transition, and either that transition can satisfy F or it cannot. If a visible transition can satisfy F no matter where the window is, then the entire sequence is correct for F.

Change minimisation With the rule-based approach, a trace is generated by successively choosing which rules to apply where in each successive state. For a formula, the definition we have so far is that every successive *pair* of states should satisfy the formula. This is not a strict enough requirement.

Recall the formula $F_0 = \mathsf{OFF}(x.s) \wedge \mathsf{ON}^\star(x.s)$. Consider the transition $\langle M_1, M_{\text{bad}} \rangle$, shown figure 1.13. In this transition, not only does $a.s$ turn ON, but the Enzyme above a disappears (the faint red Enzyme on the right is here to signal that it is missing) and the Kinase above a and b becomes connected to b (and disconnects its previous link).

The transition $\langle M_1, M_{\text{bad}} \rangle$ satisfies $F_0(a)$. Logically that is fine, but we want to stay as close the way rule-based trace generation works as possible. Since $\langle M_1, M_{\text{bad}} \rangle$ contains superfluous changes, we want to forbid it from appearing in a sequence.

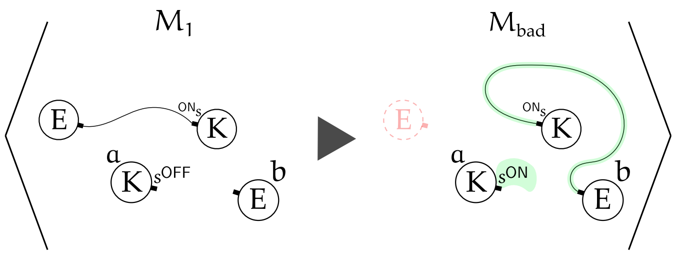


Figure 1.13.: The transition $\langle M_1, M_{bad} \rangle$ satisfies $F_0(a)$

Compare $\langle M_1, M_2 \rangle$ on figure 1.10 (p. 7) with $\langle M_1, M_{bad} \rangle$ on figure 1.13. They start from the same mixture M_1. The difference is that $\langle M_1, M_{bad} \rangle$ does strictly more than $\langle M_1, M_2 \rangle$ even though $\langle M_1, M_2 \rangle$ already satisfies $F_0(a)$.

To fix the issue, we will define a *change order* on transitions. From a same starting mixture, transitions that "do more" will be above transitions that "do less". Then we will introduce a new logical operator, $\downarrow$. With that operator, the formula $\downarrow F_0$ will only be satisfied by transitions that satisfy F_0 with *as few changes as possible*. So $\langle M_1, M_2 \rangle$ will satisfy $(\downarrow F_0)(a)$, but $\langle M_1, M_{bad} \rangle$ will not.

To faithfully reproduce the behavior of rules R_1 and R_2, the following formula can replace F:

$$F' \coloneqq (\downarrow F_1)(x) \vee (\downarrow F_2)(x, y)$$

Goals Given a formula ϕ that may use $\downarrow$ and a starting mixture, we are interested in generating all the correct traces for ϕ, and only those. Recall the rule-based trace generation process: we start from an initial mixture, find a way to match a rule to the mixture, and create a new mixture from the match. By doing it again and again, we generate a trace. Is there a process where, given a formula ϕ and an initial mixture, we can generate a sequence of new mixtures that together form a correct sequence? Can this process be complete with regards to the set of all correct sequences?

This thesis describes conditions where the answer is positive. More precisely, it starts by rephrasing the question as: *Is it possible to transform a formula into a finite set of rules such that the sequences generated by the rules are exactly those that are correct for the formula?* The rephrasing is particularly relevant because there are existing simulation engines that can take as a input a finite set of rules and efficiently output traces [Boutillier et al., 2018, Chylek et al., 2014] (they run actual simulations in the sense that their trace generation process is parameterised by stochastic rates instead of being nondeterministic). We would like to make use of them. In the next section, we provide an overview of the content of this thesis.

1.3. Plan & Contributions

The thesis is organised as follows: we start by defining transitions, rewrite rules, and how to execute those rules. Then we define a logic for specifying transitions: first-order logic with an operator ($\downarrow$) to minimise changes. We introduce a theory of transitions, a denotational notion of execution (runs), and a canonical mapping from formulas to rules.

A first set of "typing rules", $\vdash_p$, produces formulas that can be mapped to rules such that the execution of those rules generates exactly the traces denoted by the formulas.

A second set of typing rules, $\vdash_\lhd$, produces formulas that can be mapped to a finite set of rules. As a result, formulas produced by both $\vdash_\rho$ and $\vdash_\lhd$ can in principle be executed by existing efficient graph rewriting engines [Boutillier et al., 2017] modulo a reasonable theory. We provide a program syntax which always corresponds to $\vdash_\rho$, $\vdash_\lhd$-provable formulas.

The last part deals with decidability and definability issues: we provide a decidable fragment for the theory of transitions, and show when the change-minimisation operator $\downarrow$ can be eliminated.

The contents of chapters 4, 6 and 7, minus full proofs, are in [Husson and Krivine, 2019]. The main contributions of this thesis are *a)* the framework for expressing graph rewriting rules as minimised formulas *b)* the $\vdash_\rho$ deduction rules which produce formulas with infinitarily operational trace semantics, *c)* the $\vdash_\lhd$ deduction rules which, together with $\vdash_\lhd$ produce formulas with finitarily operational run semantics, and *d)* the proof that $\vdash_\rho$-provable formulas are in two very tight fragments of first-order logic, $\exists^*\forall^*$ and $\forall^*\exists^*$.

Plan in more detail:

Structures & Runs In chapter 2 we introduce *states*, a class of structures, as well as *transitions* between states. Transitions can be summarised by *patterns* and by *rules*. Given a starting state, we describe *a)* an operational notion of rule-based nondeterministic execution that associates a set of *executions* to every rule set and *b)* a denotational notion that associates a set of *runs* to every transition set. The rule-based execution method essentially transports the description of rule-based rewriting of Kappa from [Danos et al., 2012] to signatures and structures. Both runs and executions are sequences of transitions, but we use a different name for each method of generation. We then show conditions when the operational and the denotational notion coincide.

Formulas & Minimisation In chapter 3 we describe a logic on transitions. We define the semantics of formulas, both as a set of transitions and as a set of runs from initial states. We also introduce the semantics of the minimisation operator $\downarrow$, based on a notion of *change order* between transitions. Except where otherwise noted, the rest of the thesis is in FO[$\downarrow$]. Now that formulas have a sensible notion of execution, we turn to the correspondence between rule sets and formulas.

From formulas to rules Chapter 3 describes conditions when the execution semantics and the run semantics of formulas are the same. We introduce a first "type system" for formulas, $\vdash_\rho$. We show that for $\vdash_\rho$-provable formulas, there is a set of rules such that, for a class of initial states, their execution produces exactly the runs denoted by the formula.

Bounded changes The rule sets that correspond to $\vdash_\rho$-provable formulas can be infinite. Morally, the reason is that even after change minimisation the semantics of a formula does not bound the amount of change that can occur during a transition. In chapter 4 we study changes in transitions under a tighter theory, that of n-bounded linked forests, which characterises transitions between forests of bounded-height trees with functional links between tree vertices. We introduce a second "type system" for formulas, $\vdash_\lhd$, such that change-minimal models of $\vdash_\lhd$-provable formulas are bounded in their amount of change.

Compilation: from logic program to rules In chapter 5 we ask when a formula can effectively be compiled to a finite set of rules. Combining the previous chapters, we find that one answer is:

when the formula is both $\vdash_\rho$ and $\vdash_\lhd$-provable. We introduce a syntax for "programs" that compile to formulas such that any syntatically valid program compiles to a $\vdash_\rho$, $\vdash_\lhd$-provable formula. More than a toy syntax, programs show admissible rules for $\vdash_\lhd$ and $\vdash_\rho$ which demonstrates the rich syntax available for building compilable formulas.

Decidability In chapter 6 we consider decidability questions. We show that the FO theory of n-bounded-height linked forest does not have decidable satisfiability, but that its $\exists^*\forall^*$ fragment does.

Change minimality in FO In chapter 7, we introduce *unified circumscription*. Circumscription dates back to the early 1980s and is a way to characterise the minimal models of formulas along some order, using 2nd-order quantification. We show that the version we dub unified circumscription can express minimality along the change order defined in chapter 3. We then show when the minimality operator $\downarrow$ can be eliminated (thus going from FO[$\downarrow$] to FO). We further show that, modulo the theory of n-bounded-height linked forests, $\vdash_\lhd$-provable formulas are in the in the $\exists^*\forall^*$ and $\forall^*\exists^*$ fragments of FO.

2. Runs and executions for transition structures

The first chapter introduces the main notions used throughout the thesis. In section 2.1 we introduce general notations and notions. Then in section 2.2 we define states and transitions, the basic structures we use in the rest of this thesis and important notions related to them: distance, 0-embeddings, state joining. In section 2.3 we define rules and pushouts, an operational notion of trace generation (executions). Then section 2.4 gives a more denotational version of trace generation with patterns and matches (runs). Section 2.5 shows when the previous two approaches coincide for one step of trace generation. Finally, section 2.6 shows when they coincide on entire traces.

In more detail, this chapter defines *states*, which represent a snapshot of a biological mixture, and *transitions* between those states. A state is a set of elements connected in two ways. First, through *static functions*. They represent the unchanging, rigid structure of a molecule. Second, through *dynamic links*. Those typically represent temporary interactions between molecules; they are subject to change. In addition, unary properties also exist, both in static and dynamic versions. A transition therefore contains only one version of the static information, and two versions of the dynamic information: one for the state "before" the transition, and the other for "after".

Both states and transitions can be generalised to *patterns*: as in rule-based rewriting, a "small" transition can be found in a bigger one. This is called a *match*. A match from a transition to another starts with a structure embedding, but *application conditions* determine when the embedding is a proper match. This supplementary information is found in a pattern.

One can give states and transitions a notion of *execution*. First, given a rule $\mathscr{r}$ (a variation on the notion of pattern) and a state $\mathfrak{M}$, one may sometimes construct a *pushout*, which is morally a "next state" representing the result of applying $\mathscr{r}$ to a specific part of $\mathfrak{M}$. In the case of rule-based graph rewriting, this operation was shown figure 1.8 (p.5), where a rule was a pair of graphs equipped with application conditions. From now on, rules will be objects derived from states. This one-step operation can be iterated by repeated application of rules in a set to the most recent "next state", generating an execution.

A more semantic notion is that of pattern *instance*. Instead of being constructed from some "current state" $\mathfrak{M}$, an instance of $\mathscr{r}$ is any transition in which $\mathscr{r}$ *could* match. A sequence of such instances can form a *run* – the denotational counterpart to the operational notion of *execution*.

Given a set of patterns, one can derive a corresponding set of rules. The main result of this chapter gives sufficient conditions for the *runs* of a set of patterns to be equal to the *executions* of their derived rules.

A note on figures: in the sequel, figures representing structures use arrows with thick heads for the interpretation of functions and curves for binary relations; in all examples, those relations are symmetric. If a function arrow or a relation curve has no name, then the name is either obvious or unimportant. If a function arrow is not shown on some point, it is the identity there. Names such as $\mathsf{a}, \mathsf{b}, \mathsf{c}$ *are for domain elements. If for instance* a *has a subscript* A*, it means that* a *is in the interpretation of* A*.*

2.1. Generalities

In this section we describe various notations and we define the basic model theory concepts used throughout this thesis.

The marker **Notation** (as opposed to **Definition**) introduces notations for objects whose definition is outside the scope of the document, or syntactic sugar for already defined objects. The marker **Implicit parameter** introduces objects in the context of the document. They are then implicitly universally quantified.

Notation (Tuples)**.** If some symbols $(a, x, t, \ldots)$ are used to denote a certain kind of objects (variables, nonlogical constants, etc), bold symbols denote tuples of objects of that kind $(\mathbf{a}, \mathbf{x}, \mathbf{t}, \ldots)$. Set-like notations apply to tuples; for instance $(a, b) \subseteq (a, x, b, c)$ and $x \in (y, x)$. ■

Implicit parameter The universe $\mathcal{U}$ is some recursive, infinite set; the domain of a structure is always a subset of $\mathcal{U}$. An *element* always refers to an element of $\mathcal{U}$.

Basic definitions A signature Σ is a tuple of symbols. Each symbol $S \in \Sigma$ is equipped with a nature (relational or functional) and an arity $\mathrm{arity}(S) \geqslant 0$.

A Σ-structure $\mathfrak{A}$ (written $\mathfrak{A} : \Sigma$) is a domain $\mathrm{dom}(\mathfrak{A}) \subseteq \mathcal{U}$ together with an interpretation for every symbol of Σ, that is: a k-ary relation on $\mathrm{dom}(\mathfrak{A})$ (resp. k-arguments function on $\mathrm{dom}(\mathfrak{A})$) $[\![S]\!]_{\mathfrak{A}}$ for every k-ary relation symbol (resp. function symbol) $S \in \Sigma$. If $\Sigma' \subseteq \Sigma$, $[\![\Sigma']\!]_{\mathfrak{A}}$ is the tuple of interpretations of Σ'-symbols in $\mathfrak{A}$. The Σ'-reduct of $\mathfrak{A}$ is the Σ'-structure with domain $\mathrm{dom}(\mathfrak{A})$ and interpretations $[\![\Sigma']\!]_{\mathfrak{A}}$.

If $\mathfrak{B} : \Sigma$, $m : \mathrm{dom}(\mathfrak{A}) \to \mathrm{dom}(\mathfrak{B})$ is injective, and $m([\![S]\!]_{\mathfrak{A}}(\mathbf{a})) = [\![S]\!]_{\mathfrak{B}}(m(\mathbf{a}))$ (resp. $\mathbf{a} \in [\![S]\!]_{\mathfrak{A}}$ iff $m(\mathbf{a}) \in [\![S]\!]_{\mathfrak{B}}$) for every function symbol (resp. relation symbol) $S \in \Sigma$ and tuple $\mathbf{a}$ from $\mathrm{dom}(\mathfrak{A})$ of size $\mathrm{arity}(S)$, then m is an embedding from $\mathfrak{A}$ to $\mathfrak{B}$ (written $m : \mathfrak{A} \to \mathfrak{B}$).

If $\mathrm{id}_{\mathfrak{A}} : \mathfrak{A} \to \mathfrak{B}$, then $\mathfrak{A}$ is a substructure of $\mathfrak{B}$. If $K \subseteq \mathcal{U}$, the substructure of $\mathfrak{A}$ induced by K (written $\mathfrak{A} \upharpoonright K$) is the uniquely defined substructure of $\mathfrak{A}$ with $\subseteq$-minimal domain still containing $K \cap \mathrm{dom}(\mathfrak{A})$.

If $\Sigma' \subseteq \Sigma$ and $K \subseteq \mathcal{U}$, $[\![\Sigma']\!]_{\mathfrak{A}} \upharpoonright K := [\![\Sigma']\!]_{\mathfrak{A}' \upharpoonright K}$ where $\mathfrak{A}'$ is the reduct of $\mathfrak{A}$ to Σ'.

2.2. States and transitions

In this section we describe *states* and *transitions*, which are classes of structures, as well as *changes*. We then introduce 3 notions: a pseudodistance δ on structure elements that identifies functionally accessible elements, a join operation for states (when they are similar enough) that produces transitions, and a stricter notion of structure embeddings that respects the identities produced by the pseudodistance δ.

States are the *mixtures* from the introduction. Transitions represent going from one state to another in a single step. First, we describe the class of state and transitions *signatures*:

Definition (State and transition signature)**.** If

- Dyn is a tuple of relational symbols such that $|\mathsf{Dyn}| \geqslant 2$ and
  - One distinguished symbol, Link, is binary.
  - Another distinguished symbol, Has, is unary.
  - All other symbols are unary.

- $\mathsf{Dyn}^\star$ is Dyn with a star* appended to each symbol,
- Stat is a tuple of unary symbols, relational or functional,

then $\Theta := (\mathsf{Dyn}, \mathsf{Dyn}^\star, \mathsf{Stat})$ is a *transition signature* and $\Theta^\circ := (\mathsf{Dyn}, \mathsf{Stat})$ is its corresponding *state signature*. ■

Connection with rule-based models We explain the connection with the rule-based models from the introduction. Recall mixtures and transitions from the introduction: some names were static, such as K (for Kinase). A static name would be a unary predicate symbol in Stat. Other names were dynamic, such as ON. That name would be in Dyn. Its counterpart for the next mixture, $\mathsf{ON}^\star$, would be in $\mathsf{Dyn}^\star$. An agent in a mixture would be an element, say a. Its site would be another element, and the name of the site would be the name of a function $s \in \mathsf{Stat}$. So $a.s$ refers to the image of a by the interpretation of s.

So an interpretation for Stat represents the unchanging part of a structure, while there are two copies of Dyn. The interpretation of Dyn holds the *precondition*; the interpretation of $\mathsf{Dyn}^\star$ holds the *postcondition*. In a biological setting, the static information in the interpretation of Stat represents proteins names, internal protein structure, etc. The intepretations of Dyn and $\mathsf{Dyn}^\star$ hold the dynamical properties: phosphorylation state, folding, link to another protein, etc.

Has and Link We motivate the distinguished predicate and relation symbols Has, $\mathsf{Has}^\star$, Link and $\mathsf{Link}^\star$. The distinguished binary Link symbol is for protein-protein interactions. There is no technical issue with adding more relational symbols, but we have just one to keep things simple. Moreover, we single out the unary Has to encode *presence*: in a logical specification (see later), we would like to say e.g. $\mathsf{Has}(x) \wedge \neg\mathsf{Has}^\star(x)$ to ensure the destruction of x during a transition.

Notation. If Θ is a transition signature and $\mathfrak{A} : \Theta$, then $\mathfrak{A}$ is a *transition*. If $\mathfrak{M} : \Theta^\circ$, then $\mathfrak{M}$ is a *state*. The names $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}$ and variations are reserved for transitions. The names $\mathfrak{M}, \mathfrak{S}$ and variations are reserved for states ($\mathfrak{M}$ for "*m*ixture" and $\mathfrak{S}$ for "*s*tate"). ■

Definition (Function symbols of **Stat**)**.** The tuple $\mathsf{Stat}_{\mathrm{fun}} \subseteq \mathsf{Stat}$ contains exactly the functional symbols of Stat. ■

Notation. Symbols in Dyn and $\mathsf{Dyn}^\star$ are said to be *dynamic*; symbols in Stat are said to be *static*. In the sequel, the possible fonts are:

- Sans-serif for dynamic symbols.
- Regular for static symbols state/transition signature.
- *Italic* for formula names (see chapter 3).
- Math for other symbols

In any state or transition signature, any symbol $\mathsf{A} \in \mathsf{Dyn}$ has its counterpart $\mathsf{A}^\star \in \mathsf{Dyn}^\star$ of same nature and same arity. ■

Changes We introduce *changes* to capture the difference between the pre- and postconditions. The notation prefigures the use of semantic brackets for formulas.

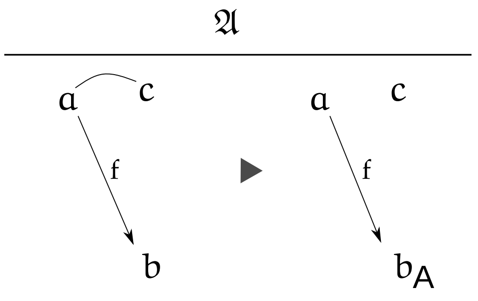
Definition (Changes)**.** If $\mathfrak{A} : \Theta$, for every $\mathsf{A} \in \mathsf{Dyn}$ let $[\![\Delta\mathsf{A}]\!]_{\mathfrak{A}} := [\![\mathsf{A}]\!]_{\mathfrak{A}} \Delta [\![\mathsf{A}^\star]\!]_{\mathfrak{A}}$, where Δ is the symmetric difference. ■

Definition (Transition changes)**.** If $\mathfrak{A} : \Theta$, let $[\![\Delta]\!]_{\mathfrak{A}}$ be the union of the carrier sets of $[\![\Delta\mathsf{A}]\!]_{\mathfrak{A}}$ for all $\mathsf{A} \in \mathsf{Dyn}$, and $[\![\neg\Delta]\!]_{\mathfrak{A}}$ its complement in $\mathfrak{A}$. ■

Implicit parameter $\Theta := (\mathbf{Dyn}, \mathbf{Dyn}^\star, \mathbf{Stat})$ is a transition signature.

Example. The transition $\mathfrak{A}$ is represented graphically figure 2.1. The static structure (interpretation of $\mathbf{Stat}$) is repeated left and right of the arrow ▶. The interpretation of $\mathbf{Dyn}$ is left of the arrow: there is a symmetric Link between a and c. The interpretation of $\mathbf{Dyn}^\star$ is right of the arrow: the Link between a and c is gone, and b has gained the $\mathsf{A}^\star$ property.

If the image of some function on an element is not shown, it implicitly means that the function is the identity at that point. For instance, $[\![f]\!]_{\mathfrak{A}}(b) = b$. ■



$$\begin{aligned}
\mathbf{Dyn} &= (\mathsf{A}, \mathsf{Link})\\
\mathbf{Dyn}^\star &= (\mathsf{A}^\star, \mathsf{Link}^\star)\\
\mathbf{Stat}^\star &= (f)\\
[\![\mathsf{A}]\!]_{\mathfrak{A}} &= \emptyset\\
[\![\mathsf{A}^\star]\!]_{\mathfrak{A}} &= \{b\}\\
[\![\Delta]\!]_{\mathfrak{A}} &= \{a, b, c\}\\
[\![\Delta\mathsf{Link}]\!]_{\mathfrak{A}} &= \{(a, c), (c, a)\}
\end{aligned}$$

Figure 2.1.: Transition $\mathfrak{A}$.

The next three subsections introduce concepts important to states and transitions: distance, 0-embeddings, and how to create transitions from states. The distance treats paths through $\mathbf{Stat}_{\text{fun}}$ differently from paths through $\mathsf{Link}, \mathsf{Link}^\star$. 0-embeddings relate two structures by defining when one can be “found” in the other. Joins trivially construct a transition from two states, but they are not always well-defined.

2.2.1. Pseudodistance

In this subsection we define a pseudodistance between structure elements.

In the sequel, many properties will depend on identifying a small part of a state or transition which should satisfy some constraint, and beyond which constraints are looser. As we will see later, the difficulty comes from the dynamic part of the structures, because static structures are always conserved through morphisms (see next subsection) and when necessary become bounded in size (see chapter 4). So we will define a notion of pseudodistance which identifies all elements *statically* connected and only counts as a “far away” elements which may become disconnected from one another. In a state or a transition, function symbols are only in $\mathbf{Stat}_{\text{fun}}$; they represent immutable structure of a mixture. On the other hand, links (Link and $\mathsf{Link}^\star$) are dynamic: they can appear and disappear. So we define a notion of pseudodistance which ignores functions:

Definition (Pseudodistance δ)**.** If $\mathfrak{A} : \Theta$ or $\mathfrak{A} : \Theta^\circ$, let $G_{\mathfrak{A}}$ be the undirected, weighted graph with vertices $\mathrm{dom}(\mathfrak{A})$ and, for every $a, b \in \mathrm{dom}(\mathfrak{A})$:

- an edge $\{a, b\}$ of weight 0 whenever $[\![h]\!]_{\mathfrak{A}}(a) = b$ for some $h \in \mathbf{Stat}_{\text{fun}}$,
- otherwise, an edge $\{a, b\}$ of weight 1 whenever $(a, b) \in [\![R]\!]_{\mathfrak{A}}$ for some binary, relational $R \in \Theta$ or Θ°.

The following function is a pseudo[1]distance: for every $a, b \in \mathrm{dom}(\mathfrak{A})$, $\delta_{\mathfrak{A}}(a, b)$ is the length of

[1] $\delta(a, b) = 0$ does not imply $a = b$.

the shortest a—b path in G. ■

Balls Taking balls of radius d around elements or subdomains of structures will be very useful in the sequel. However, note that the ball of radius 0 is particularly meaningful here since δ is a pseudodistance: for an element a in a structure, the set of elements at distance 0 (its 0-neighbors) are the elements functionally and inverse-functionally accessible from a.

Definition (Balls). If $\mathfrak{A} : \Theta$ of $\mathfrak{A} : \Theta^{\circ}$, $d \geqslant 0$ and $a \in \mathrm{dom}(\mathfrak{A})$, the *ball of radius* d *around* a is

$$\mathcal{B}_{\mathfrak{A}}(a, d) := \{ b \in \mathfrak{A} \mid \delta_{\mathfrak{A}}(a, b) \leqslant d \}$$

and naturally extends to subsets of $\mathcal{U}$: if $K \subseteq \mathcal{U}$, $\mathcal{B}_{\mathfrak{A}}(K, d)$ is the union of the balls of radius d around each $a \in K \cap \mathrm{dom}(\mathfrak{A})$. ■

An important feature of δ is that elements that are *statically* connected are considered "the same" from a distance perspective.

Example. In figure 2.2, the state $\mathfrak{S}$ has 4 elements. b is symetrically Linked to d. The names of the functions in $\mathbf{Stat}_{\mathrm{fun}}$ are not shown, we only see arrows, indicating that for some $f_1, f_2 \in \mathbf{Stat}_{\mathrm{fun}}$, $[\![f_1]\!]_{\mathfrak{S}}(a) = c$ and $[\![f_2]\!]_{\mathfrak{S}}(a) = b$. As earlier, function images not shown implicitly mean a loop. For instance, $[\![f_2]\!]_{\mathfrak{S}}(b) = b$. The corresponding graph $G_{\mathfrak{S}}$ is shown on the right, and the ball of radius 0 around b is shown in a semitransparent blue outline. ■

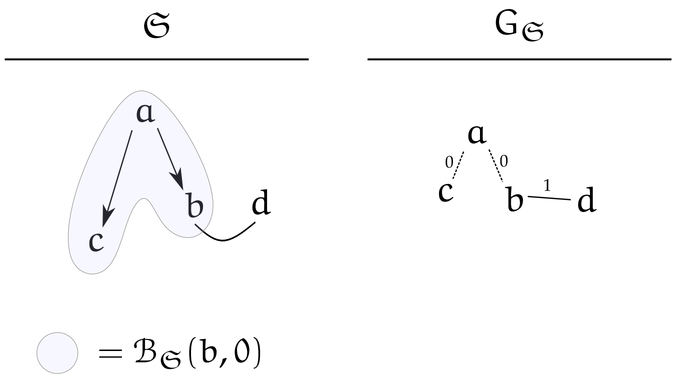


Figure 2.2.: State $\mathfrak{S}$ and distance between its elements.

Using the above notion of distance, it is possible to say when two states are similar enough to be joined into a transition structure.

2.2.2. Joining states into transitions

In this subsection we give the canonical way to assemble states into transitions. The operation is very simple, and the side-condition ensures there is no clash between the static parts of each state. We then give a simple lemma which relates distance in joined states and in each individual state.

For any two states $\mathfrak{S}$, $\mathfrak{S}'$, both of their domains are in a common universe $\mathcal{U}$. We forgo looking for partial isomorphisms between states and simply consider the common elements of $\mathrm{dom}(\mathfrak{S})$ and $\mathrm{dom}(\mathfrak{S}')$. If the static structure (that is, the 0-ball) around them is the same, then $\mathfrak{S}$ and $\mathfrak{S}'$ are "similar enough" to be meaningfully joined into a new transition:

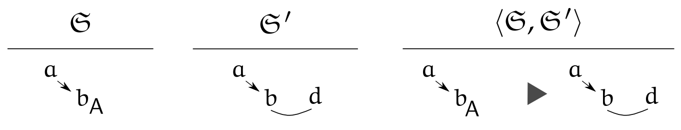


Figure 2.3.: $\langle \mathfrak{S}, \mathfrak{S}' \rangle$ is well-defined.

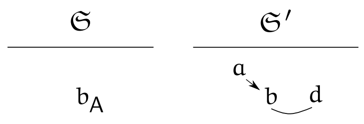


Figure 2.4.: $\langle \cdot, \cdot \rangle$ not defined on $\mathfrak{S}, \mathfrak{S}'$

Definition ($\langle \cdot, \cdot \rangle$). If $\mathfrak{S}, \mathfrak{S}' : \Theta^\circ$ and $[\![\mathbf{Stat}]\!]_{\mathfrak{S}} \restriction \mathcal{B}_{\mathfrak{S}}(\mathrm{dom}(\mathfrak{S}'), 0) = [\![\mathbf{Stat}]\!]_{\mathfrak{S}'} \restriction \mathcal{B}_{\mathfrak{S}'}(\mathrm{dom}(\mathfrak{S}), 0)$, then $\langle \mathfrak{S}, \mathfrak{S}' \rangle : \Theta$ is:

$$\begin{aligned}
\mathrm{dom}(\langle \mathfrak{S}, \mathfrak{S}' \rangle) &:= \mathrm{dom}(\mathfrak{S}) \cup \mathrm{dom}(\mathfrak{S}') \\
[\![A]\!]_{\langle \mathfrak{S}, \mathfrak{S}' \rangle} &:= [\![A]\!]_{\mathfrak{S}} && \text{for all } A \in \mathbf{Dyn} \\
[\![A^\star]\!]_{\langle \mathfrak{S}, \mathfrak{S}' \rangle} &:= [\![A]\!]_{\mathfrak{S}'} && \text{for all } A \in \mathbf{Dyn} \\
[\![T]\!]_{\langle \mathfrak{S}, \mathfrak{S}' \rangle} &:= [\![T]\!]_{\mathfrak{S}} \cup [\![T]\!]_{\mathfrak{S}'} && \text{for all } T \in \mathbf{Stat}
\end{aligned}$$

$\langle \mathfrak{S}, \mathfrak{S}' \rangle$ is well-defined thanks to the proviso on $[\![\mathbf{Stat}]\!]_{\mathfrak{S}}$ and $[\![\mathbf{Stat}]\!]_{\mathfrak{S}'}$. ■

Example (and counterexample)**.** The states $\mathfrak{S}$ and $\mathfrak{S}'$ in figure 2.3 respect the conditions necessary for $\langle \mathfrak{S}, \mathfrak{S}' \rangle$ to be well-defined.

On the other hand, in figure 2.4, $\langle \mathfrak{S}, \mathfrak{S}' \rangle$ is not defined because $[\![f]\!]_{\mathfrak{S}} \restriction \{a\} = \{(a,a)\}$ and $[\![f]\!]_{\mathfrak{S}'} \restriction \{a,b\} = \{(a,a),(b,a)\}$. ■

Formally, the condition $[\![\mathbf{Stat}]\!]_{\mathfrak{S}} \restriction \mathcal{B}_{\mathfrak{S}}(\mathrm{dom}(\mathfrak{S}'), 0) = [\![\mathbf{Stat}]\!]_{\mathfrak{S}'} \restriction \mathcal{B}_{\mathfrak{S}'}(\mathrm{dom}(\mathfrak{S}), 0)$ is more than is necessary for $\langle \mathfrak{S}, \mathfrak{S}' \rangle$ to be well-defined. It would suffice to require $[\![\mathbf{Stat}]\!]_{\mathfrak{S}} \restriction \mathfrak{S}' = [\![\mathbf{Stat}]\!]_{\mathfrak{S}'} \restriction \mathfrak{S}$. But then joins such as in figure 2.4 would be allowed. The issue with these joins is that static structures are supposed to *not* change from one state to the next. The creation of d and of a b–d link between $\mathfrak{S}$ and $\mathfrak{S}'$ is allowed because presence and linkage are dynamic properties. But the appearance of an a in the function graph of b violates the design rule about static functions and predicates. More formally, the information loss about the structure of $\mathfrak{S}$ (in figure 2.4) would prevent later operations that extract states back from transitions; those operations use the lemma we are just about to state, which becomes false if we use the looser requirement.

We explain that next lemma. After a join, the distance between two elements can become smaller, for instance by the addition of a link. It is possible for two elements that were infinitely far away in two states $\mathfrak{S}$ and $\mathfrak{S}'$ to be at finite distance in $\langle \mathfrak{S}, \mathfrak{S}' \rangle$. However (keeping with the spirit of static functions being really static), the 0 bound is strict: elements at distance > 0 in a state cannot be at distance 0 after a join.

Lemma 2.2.1. If $\langle \mathfrak{M}, \mathfrak{M}' \rangle$ is defined, $\delta_{\langle \mathfrak{M}, \mathfrak{M}' \rangle}(a,b) = 0$ and $a \in \mathrm{dom}(\mathfrak{M})$, (resp $a \in \mathrm{dom}(\mathfrak{M}')$) then $\delta_{\mathfrak{M}}(a,b) = 0$ (resp. $\delta_{\mathfrak{M}'}(a,b) = 0$).

Proof. We show for $\mathfrak{M}$, the other version is symmetric.

First, take any $a', b' \in \mathrm{dom}(\langle \mathfrak{M}, \mathfrak{M}' \rangle)$ such that $(a',b') \in [\![h]\!]_{\langle \mathfrak{M}, \mathfrak{M}' \rangle}$ for some function symbol $h \in \mathbf{Stat}$. We show that if a' or b' is in $\mathrm{dom}(\mathfrak{M})$, the other is as well, and $(a',b') \in [\![h]\!]_{\mathfrak{M}}$.

By definition of $[\![h]\!]_{\langle\mathfrak{M},\mathfrak{M}'\rangle}$, the interesting case is $(a',b') \in ([\![h]\!]_{\mathfrak{M}'})$; if one of a',b' is in $\mathfrak{M}$, say a' for concreteness, then since $[\![\mathbf{Stat}]\!]_{\mathfrak{M}} \restriction \mathcal{B}_{\mathfrak{M}}(\mathrm{dom}(\mathfrak{M}'),0) = [\![\mathbf{Stat}]\!]_{\mathfrak{M}'} \restriction \mathcal{B}_{\mathfrak{M}'}(\mathrm{dom}(\mathfrak{M}),0)$, $(a',b') \in [\![h]\!]_{\mathfrak{M}}$ as well.

By the above and by induction on the number of 0-edges of any a–b path in $G_{\langle\mathfrak{M},\mathfrak{M}'\rangle}$, we get $a \in \mathrm{dom}(\mathfrak{M})$ and $\delta_{\mathfrak{M}}(a,b) = 0$. □

We will make use of the operation $\langle\cdot,\cdot\rangle$ throughout this chapter and the next.

2.2.3. 0-embedding

In this subsection we define a restriction of embeddings called 0-embeddings and show that 0-embeddings compose.

0-embeddings are the basis for defining when a small transition can be “found” in a bigger one. They follow the mantra started earlier: elements at distance 0 are immutably connected, so a morphism should not be allowed to give new 0-neighbors to an element.

0-embeddings are not powerful enough, however: we will later introduce the concept of a match, which refines 0-embeddings further by adding *application conditions*, as seen in the introduction, of the form “the image of this element may not gain new 1-neighbors”. Unlike the constraint of 0-embeddings on the absence of new 0-neighbors, which is universal, matches will specify which elements cannot gain new 1-neighbors.

Definition (0-embedding)**.** If $\mathfrak{A},\mathfrak{B} : \Theta$ or $\mathfrak{A},\mathfrak{B} : \Theta^{\circ}$, an embedding $m : \mathfrak{A} \to \mathfrak{B}$ is a 0-*embedding* if

$$\mathcal{B}_{\mathfrak{B}}(m(\mathfrak{A}),0) \subseteq m(\mathfrak{A})$$

■

A 0-embedding is simply an embedding that respects the pseudodistance δ's identities. An alternative design would have been to only consider the class of structures where the following is true: for every pair of elements a,b, b accessible from a in the function graph implies a accessible from b in the function graph. For such structures, all embeddings are 0-embeddings.

Example (Counterexample)**.** In figure 2.5, the identity $\mathrm{id}_{\mathfrak{S}}$ is an embedding but it is *not* a 0-embedding since the 0-ball around b in $\mathfrak{M}$ is not part of $\mathrm{Im}(\mathrm{id}_{\mathfrak{A}})$. ■

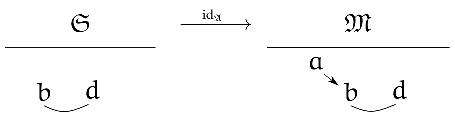


Figure 2.5.: $\mathrm{id}_{\mathfrak{A}} : \mathfrak{S} \to \mathfrak{M}$

The following lemma illustrates the action of a 0-embedding (note that it does not hold for embeddings):

Lemma 2.2.2. If $m : \mathfrak{A} \to \mathfrak{B}$ is a 0-embedding, $a \in \mathrm{Im}(m)$ and $\delta_{\mathfrak{B}}(a,b) = 0$, then $b \in \mathrm{Im}(m)$ and $\delta_{\mathfrak{B}}(m^{-1}(a), m^{-1}(b)) = 0$

Proof. Since $\mathcal{B}_{\mathfrak{B}}(\mathrm{Im}(m),0) \subseteq \mathrm{Im}(m)$, $b \in \mathrm{Im}(m)$ and by induction on the number of 0-edges in any path a–b in $G_{\mathfrak{B}}$, since m is an embedding $\delta_{\mathfrak{A}}(m^{-1}(a), m^{-1}(b)) = 0$. □

Lemma 2.2.2 immediately implies composition of 0-embeddings:

Lemma 2.2.3. If $\mathfrak{m} : \mathfrak{A} \to \mathfrak{B}$ and $\mathfrak{m}' : \mathfrak{B} \to \mathfrak{C}$ are both 0-embeddings, then $\mathfrak{m}' \circ \mathfrak{m}$ is a 0-embedding as well.

Proof. Immediate by lemma 2.2.2. □

With the above ingredients (state joining, 0-embeddings and distance), it is possible to define rules and how to apply them to states.

2.3. Rules, pushouts, executions

Now that states, transitions, and the morphisms between them are established, it is time to define our structure-based version of rule application in graph rewriting. In this section, we define those and show that a rule applied to a *current* state always creates a *next* state which can be joined to the current one.

Consider two states $\mathfrak{S}, \mathfrak{S}'$ and another state $\mathfrak{M}$ such that it makes sense to talk about $\langle \mathfrak{S}, \mathfrak{S}' \rangle$ and such that $\mathfrak{S}$ can in some sense be found in $\mathfrak{M}$. We want to *transport the differences between* $\mathfrak{S}$ *and* $\mathfrak{S}'$ *to* $\mathfrak{M}$, thus creating a new state $\mathfrak{M}'$. The operation defined below is meant to be an idealised version of how graph rewriting frameworks such as Kappa work.

Definition (Pushout)**.** If $\mathfrak{S}, \mathfrak{S}', \mathfrak{M} : \Theta^\circ$, $\langle \mathfrak{S}, \mathfrak{S}' \rangle$ is defined, $f : \mathfrak{S} \to \mathfrak{M}$ is a 0-embedding, $\mathcal{B}_{\mathfrak{M}}(f(\mathrm{dom}(\mathfrak{S}) \setminus \mathrm{dom}(\mathfrak{S}')), 1) \subseteq \mathrm{Im}(f)$ and $g : \mathrm{dom}(\mathfrak{S}') \setminus \mathrm{dom}(\mathfrak{S}) \to \mathfrak{U} \setminus \mathrm{dom}(\mathfrak{M})$ is injective, then the *pushout* of $(f, g, \mathfrak{S}, \mathfrak{S}', \mathfrak{M})$ is $\mathfrak{M}'$:

$$\begin{aligned}
\mathrm{dom}(\mathfrak{M}') &:= \mathrm{dom}(\mathfrak{M}) \setminus f(\mathrm{dom}(\mathfrak{S}) \setminus \mathrm{dom}(\mathfrak{S}')) \uplus \mathrm{Im}(g) & \\
[\![A]\!]_{\mathfrak{M}'} &:= [\![A]\!]_{\mathfrak{M}} \setminus f([\![A]\!]_{\mathfrak{S}} \setminus [\![A]\!]_{\mathfrak{S}'}) \uplus (f \uplus g)([\![A]\!]_{\mathfrak{S}'} \setminus [\![A]\!]_{\mathfrak{S}}) & \text{for all } A \in \mathbf{Dyn} \\
[\![T]\!]_{\mathfrak{M}'} &:= [\![T]\!]_{\mathfrak{M}} \upharpoonright \mathrm{dom}(\mathfrak{M}') \cup (f \uplus g)([\![T]\!]_{\mathfrak{S}'}) & \text{for all } T \in \mathbf{Stat}
\end{aligned}$$

It is well-defined. First, for all $T \in \mathbf{Stat}$ $[\![T]\!]_{\mathfrak{M}} \upharpoonright \mathrm{dom}(\mathfrak{M}')$ and $(f \uplus g)([\![T]\!]_{\mathfrak{S}'})$ agree on their intersection because $\langle \mathfrak{S}, \mathfrak{S}' \rangle$ is defined and the image of g does not intersect $\mathrm{dom}(\mathfrak{M})$. Next, the domain of $\mathfrak{M}'$ should cover the interpretation of all symbols in Θ°:

- If $\mathbf{a} \in [\![A]\!]_{\mathfrak{M}}$ and for some $a \in \mathbf{a}$, $a \in f(\mathrm{dom}(\mathfrak{S}) \setminus \mathrm{dom}(\mathfrak{S}'))$ then by $\mathcal{B}_{\mathfrak{M}}(f(\mathrm{dom}(\mathfrak{S}) \setminus \mathrm{dom}(\mathfrak{S}')), 1) \subseteq \mathrm{Im}(f)$, and the fact that f is an embedding, $\mathbf{a} \in f([\![A]\!]_{\mathfrak{S}})$. Since $f^{-1}(a) \notin \mathrm{dom}(\mathfrak{S})$, $f^{-1}(\mathbf{a}) \notin [\![A]\!]_{\mathfrak{S}'}$, and so $\mathbf{a} \notin [\![A]\!]_{\mathfrak{M}'}$.

- Suppose there is $\mathbf{a} \in [\![A]\!]_{\mathfrak{S}'} \setminus [\![A]\!]_{\mathfrak{S}}$. For every $a \in \mathbf{a}$, a is not in $\mathrm{dom}(\mathfrak{S}) \setminus \mathrm{dom}(\mathfrak{S}')$, moreover if $(f \uplus g)(a) \notin \mathrm{dom}(\mathfrak{M})$ then $a \in \mathrm{dom}(\mathfrak{S}') \setminus \mathrm{dom}(\mathfrak{S})$ by assumption on f, g.

- For the relational symbols of $\mathbf{Stat}$, trivial. We show that if $h \in \mathbf{Stat}_{\mathrm{fun}}$, $a \in \mathrm{dom}(\mathfrak{M}')$ and $(a, b) \in [\![h]\!]_{\mathfrak{M}'}$, then $b \in \mathrm{dom}(\mathfrak{M}')$:
  - If $(a, b) \in (f \uplus g)([\![T]\!]_{\mathfrak{S}'})$ then $b \in (f \uplus g)(\mathrm{dom}(\mathfrak{S}'))$. If $b \in \mathrm{Im}(g)$, $b \in \mathrm{dom}(\mathfrak{M}')$ by definition of $\mathrm{dom}(\mathfrak{M}')$, otherwise $b \in \mathrm{Im}(f)$, so $b \in \mathrm{dom}(\mathfrak{M})$ by definition of f, and $b \notin f(\mathrm{dom}(\mathfrak{S}) \setminus \mathrm{dom}(\mathfrak{S}'))$, so $b \in \mathrm{dom}(\mathfrak{M}')$ by definition of $\mathrm{dom}(\mathfrak{M}')$.
  - If $(a, b) \in [\![T]\!]_{\mathfrak{M}} \upharpoonright \mathrm{dom}(\mathfrak{M}')$ then $b \in \mathrm{dom}(\mathfrak{M})$, so we need to show $b \notin f(\mathrm{dom}(\mathfrak{S}) \setminus \mathrm{dom}(\mathfrak{S}'))$. Suppose $b \in \mathrm{Im}(f)$. Since f is a 0-embedding, $a \in \mathrm{Im}(f)$ and $(f^{-1}(a), f^{-1}(b)) \in [\![h]\!]_{\mathfrak{S}}$. Since $a \in \mathrm{dom}(\mathfrak{M}')$, by definition of $\mathrm{dom}(\mathfrak{M}')$ we have $f^{-1}(a) \in \mathrm{dom}(\mathfrak{S}')$ (otherwise a would be in $f(\mathrm{dom}(\mathfrak{S}) \setminus \mathrm{dom}(\mathfrak{S}'))$. So by assumption on $\langle \mathfrak{S}, \mathfrak{S}' \rangle$ (that is, $[\![\mathbf{Stat}]\!]_{\mathfrak{S}} \upharpoonright \mathrm{dom}(\mathfrak{S}') = [\![\mathbf{Stat}]\!]_{\mathfrak{S}'} \upharpoonright \mathrm{dom}(\mathfrak{S})$) $(f^{-1}(a), f^{-1}(b)) \in [\![h]\!]_{\mathfrak{S}'}$. So $f^{-1}(b) \in \mathrm{dom}(\mathfrak{S}')$, so $b \notin f(\mathrm{dom}(\mathfrak{S}) \setminus \mathrm{dom}(\mathfrak{S}'))$.

■

The definition above realizes literally transports the differences between $\mathfrak{G}$ and $\mathfrak{G}'$ along f and g, and applies those to $\mathfrak{M}$ in order to construct $\mathfrak{M}'$. Morally, $\mathfrak{M}'$ is to $\mathfrak{M}$ what $\mathfrak{G}'$ is to $\mathfrak{G}$.

Design note. To illustrate why the proviso $\mathcal{B}_{\mathfrak{M}}(f(\mathrm{dom}(\mathfrak{G}) \setminus \mathrm{dom}(\mathfrak{G}'), 1) \subseteq \mathrm{Im}(f)$ is necessary, consider some $a \in \mathrm{dom}(\mathfrak{G}) \setminus \mathrm{dom}(\mathfrak{G}')$ such that there is $b \in \mathrm{dom}(\mathfrak{M}) \setminus \mathrm{Im}(f)$ and $(f(a), b) \in [\![\mathsf{Link}]\!]_{\mathfrak{M}}$. In other words: a "disappears" in $\langle \mathfrak{G}, \mathfrak{G}' \rangle$, yet at the same time the f-image of a has a neighbor in $\mathfrak{M}$. Here the definition of $\mathfrak{M}'$ would leave a out of $\mathrm{dom}(\mathfrak{M}')$ but keep $(f(a), b)$ in $[\![\mathsf{Link}]\!]_{\mathfrak{M}'}$, which would result in an ill-defined structure.

We believe that it may be possible to have a more complex pushout definition and get rid of the proviso. The more complex definition would discard atoms that mention disappearing elements (such as the pair $(f(a), b)$). As a point of interest, this seems related to the difference between forms of graph rewriting which can allow side-effects (SPO) or forbid them (DPO). ■

Example. This is an example of a pushout. In figure 2.6, $\mathrm{id}_{\mathfrak{G}} : \mathfrak{G} \to \mathfrak{M}$ is a 0-embedding. $\mathrm{dom}(\mathfrak{G}) \setminus \mathrm{dom}(\mathfrak{G}') = \{c\}$, and c does not get any new link when going to $\mathfrak{M}$. $g = \mathrm{id}_{\{e\}}$ maps the new element e of $\mathfrak{G}'$. Since $b \in [\![A]\!]_{\mathfrak{G}}$ but $b \notin [\![A]\!]_{\mathfrak{G}'}$, $b \in [\![A]\!]_{\mathfrak{M}}$ but $b \notin [\![A]\!]_{\mathfrak{M}'}$. ■

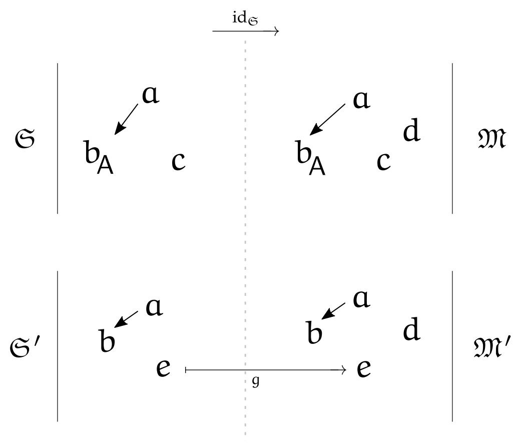


Figure 2.6.: $\mathfrak{M}'$ is the pushout of $(\mathrm{id}_{\mathfrak{G}}, g, \mathfrak{G}, \mathfrak{G}', \mathfrak{M})$.

This next lemma shows that the pushout construction (which correspond to rule application, as we will see later) creates a *next* state similar enough to the *current* one for the join operation to be well-defined:

Lemma 2.3.1. If $\mathfrak{M}'$ is the pushout of $(f, g, \mathfrak{G}, \mathfrak{G}', \mathfrak{M})$ then $\langle \mathfrak{M}, \mathfrak{M}' \rangle$ is defined.

Proof. We want $[\![\mathbf{Stat}]\!]_{\mathfrak{M}} \restriction \mathcal{B}_{\mathfrak{M}}(\mathrm{dom}(\mathfrak{M}'), 0) = [\![\mathbf{Stat}]\!]_{\mathfrak{M}'} \restriction \mathcal{B}_{\mathfrak{M}'}(\mathrm{dom}(\mathfrak{M}), 0)$. Let $m := f \uplus g$. First, note that since g does not touch $\mathfrak{M}$, and $\mathrm{dom}(\mathfrak{G}) \cap \mathrm{dom}(\mathfrak{G}') \subseteq \mathcal{B}_{\mathfrak{G}'}(\mathrm{dom}(\mathfrak{G}), 0)$, the following holds:

$$\mathrm{dom}(\mathfrak{M}) \cap m(\mathrm{dom}(\mathfrak{G}')) \subseteq m(\mathcal{B}_{\mathfrak{G}'}(\mathrm{dom}(\mathfrak{G}), 0)) \tag{2.1}$$

Moreover g defined on $\mathfrak{G}' \setminus \mathfrak{G}$ and $\mathcal{B}_{\mathfrak{G}'}(\mathrm{dom}(\mathfrak{G}), 0) \subseteq \mathrm{dom}(\mathfrak{G})$ provide the $\supseteq$-direction in the following equality (the other direction is trivial):

$$f\left([\![T]\!]_{\mathfrak{G}'} \restriction \mathcal{B}_{\mathfrak{G}'}(\mathrm{dom}(\mathfrak{G}), 0)\right) = m([\![T]\!]_{\mathfrak{G}'}) \restriction m(\mathcal{B}_{\mathfrak{G}'}(\mathrm{dom}(\mathfrak{G}), 0)) \tag{2.2}$$

(note that the left-hand side of (2.2) is well-defined since any ball is closed under function application for every function of a structure).

Now:

$$\begin{aligned}
&[\![T]\!]_{\mathfrak{M}} \upharpoonright \mathcal{B}_{\mathfrak{M}}(\mathrm{dom}(\mathfrak{M}'), 0) && \\
&= f([\![T]\!]_{\mathfrak{G}} \upharpoonright \mathcal{B}_{\mathfrak{G}}(\mathrm{dom}(\mathfrak{G}'), 0)) \cup [\![T]\!]_{\mathfrak{M}} \upharpoonright \mathcal{B}_{\mathfrak{M}}(\mathrm{dom}(\mathfrak{M}'), 0) && \text{since } f \text{ is a } 0\text{-embedding} \\
&= f([\![T]\!]_{\mathfrak{G}'} \upharpoonright \mathcal{B}_{\mathfrak{G}'}(\mathrm{dom}(\mathfrak{G}), 0)) \cup [\![T]\!]_{\mathfrak{M}} \upharpoonright \mathcal{B}_{\mathfrak{M}}(\mathrm{dom}(\mathfrak{M}'), 0) && \text{since } \langle \mathfrak{G}, \mathfrak{G}' \rangle \text{ is defined} \\
&= m([\![T]\!]_{\mathfrak{G}'}) \upharpoonright m(\mathcal{B}_{\mathfrak{G}'}(\mathrm{dom}(\mathfrak{G}), 0)) \cup [\![T]\!]_{\mathfrak{M}} \upharpoonright \mathrm{dom}(\mathfrak{M}') && \text{by (2.2) (left) and trivially (right)} \\
&= m([\![T]\!]_{\mathfrak{G}'}) \upharpoonright \mathrm{dom}(\mathfrak{M}) \cup [\![T]\!]_{\mathfrak{M}} \upharpoonright \mathrm{dom}(\mathfrak{M}') && \text{by (2.1)} \\
&= \left(m([\![T]\!]_{\mathfrak{G}'}) \cup [\![T]\!]_{\mathfrak{M}} \upharpoonright \mathrm{dom}(\mathfrak{M}')\right) \upharpoonright \mathrm{dom}(\mathfrak{M}) && \text{trivial} \\
&= [\![T]\!]_{\mathfrak{M}'} \upharpoonright \mathrm{dom}(\mathfrak{M}) && \text{by definition of } [\![T]\!]_{\mathfrak{M}'}
\end{aligned}$$

□

A rule contains the data necessary to build a pushout plus some *additional* information:

Definition (Rule). A *rule* $\mathscr{r}$ is a triple of the form $(\mathfrak{G}, K, \mathfrak{G}')$ such that $\mathfrak{G}, \mathfrak{G}' : \Theta^\circ$ and $K \subseteq \mathfrak{G}$. ■

The K part, called the application condition of a rule, is a form of *application condition* from graph rewriting. When a rule (in the graph rewriting sense) is applied to a graph, the larger context may add edges between nodes present in the rule and nodes from the context. To forbid that, one can specify nodes that should be *free*, i.e. nodes that should not gain edges by going into a larger context. Given a 0-embedding $m : \mathfrak{A} \to \mathfrak{B}$ between two structures with a defined distance, an element $a \in \mathrm{dom}(\mathfrak{A})$ gains no edges (no 1-neighbor) if the following is true :

$$\mathcal{B}_{\mathfrak{B}}(m(a), 1) \subseteq \mathrm{Im}(m)$$

Note that since m is an embedding, new edges must touch the context, that is: the complement of the image of m.

We define what a rule application is and explain more afterwards:

Definition (Rule(s) application). If $\mathscr{r} := (\mathfrak{G}, K, \mathfrak{G}')$ is a rule and $\mathfrak{M}, \mathfrak{M}' : \Theta^\circ$ then $\mathfrak{M} \twoheadrightarrow_{\mathscr{r}} \mathfrak{M}'$ if there are f, g such that $\mathfrak{M}'$ is the pushout of $(f, g, \mathfrak{G}, \mathfrak{G}', \mathfrak{M})$ and $\mathcal{B}_{\mathfrak{M}}(f(K), 1) \subseteq \mathrm{Im}(f)$.

In that case, f, g are *pushout witnesses* for $\mathfrak{M} \twoheadrightarrow_{\mathscr{r}} \mathfrak{M}'$.

If R is a set of rules, then $\mathfrak{M} \twoheadrightarrow_R \mathfrak{M}'$ if $\mathfrak{M} \twoheadrightarrow_{\mathscr{r}} \mathfrak{M}'$ for some $\mathscr{r} \in R$

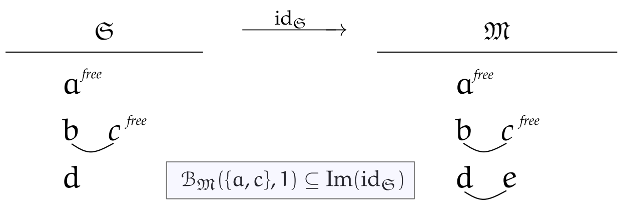
■

Example. This example shows how to apply the left-hand side of a rule to a state. As illustrated in figure 2.7, the identity on $\mathfrak{G}$ is a good candidate for a successful application of the rule $(\mathfrak{G}, \{a, c\}, \mathfrak{G}')$ (for some $\mathfrak{G}'$ not given here). $d \notin \{a, c\}$ so the new link between d and e is not a problem. Note that c is free even though it has a link; the issue is newly formed links, not existing one (in that sense our notion of *free* is different from the usual one in graph rewriting).

■

We have formalised the rules and rule applications given in the introduction. In particular, a rule $(\mathfrak{G}, K, \mathfrak{G}')$ must find a 0-embedding between its left-hand side $\mathfrak{G}$ and a state $\mathfrak{M}$, with the application condition $\mathcal{B}_{\mathfrak{A}}(m(K), 1) \subseteq m(\mathrm{dom}(\mathfrak{G}))$. In the language of rule-based models, the rule is said to have a *match* in the larger mixture $\mathfrak{M}$.

However, formulas (introduced later) specify sets of *transitions*. In the next section, we define how the notion of *match* can be defined between transitions. We will later link that denotational notion to the rule application we have just defined.



$\mathfrak{S}$
$\mathrm{id}_{\mathfrak{S}}$
$\mathfrak{M}$
a^{free}
b
c^{free}
d
e
$\mathcal{B}_{\mathfrak{M}}(\{a,c\},1) \subseteq \mathrm{Im}(\mathrm{id}_{\mathfrak{S}})$

Figure 2.7.: $\mathrm{id}_{\mathfrak{S}} : \mathfrak{S} \to \mathfrak{M}$

2.4. Patterns and matches

In this section, we introduce patterns, which are transitions equipped with *application conditions* (just as rules had an application in the previous section). Then, the notion of 0-embedding is refined to that of match: a match from a pattern is just a 0-embedding that respects the application conditions of the pattern. The set of transitions where a pattern p can match is called the set of instances of p.

We also give some simple lemmas and show that matches compose.

Patterns

Definition (Pattern)**.** If Σ is a signature, $\mathfrak{A} : \Sigma$ and $K \subseteq \mathrm{dom}(\mathfrak{A})$, the pair a $(\mathfrak{A}, K)$ is a *pattern*. $(\mathfrak{A}, K) : \Sigma$ means that $(\mathfrak{A}, K)$ is a pattern and $\mathfrak{A} : \Sigma$. ■

A stricter notion of pattern is sometimes necessary because of the proviso $\mathcal{B}_{\mathfrak{M}}(f(\mathrm{dom}(\mathfrak{S}) \setminus \mathrm{dom}(\mathfrak{S}'), 1) \subseteq \mathrm{Im}(f)$ in the definition of a pushout. If an element a in a pattern $(\langle \mathfrak{S}, \mathfrak{S}' \rangle, K)$ disappears and has gained an edge after a morphism m into a transition $\langle \mathfrak{M}, \mathfrak{M}' \rangle$, the proviso will not be respected (here f will be $m \restriction \mathrm{dom}(\mathfrak{S})$) and $\mathfrak{M}'$ will not be a pushout. Pure patterns avoid the issue – and the patterns we will most often manipulate later on will all be pure by construction.

Definition (Pure pattern)**.** If $(\mathfrak{A}, K)$ is a pattern and $[\![\Delta]\!]_{\mathfrak{A}} \subseteq K$, then $(\mathfrak{A}, K)$ is a *pure* pattern. ■

At this point, it is reasonable to ask why we define a notion so similar to rules. Rules and pushouts are operational, and should represent an idealised version of how actual rewriting programs work. But we will later reason using the denotational notion of pattern and match. In the denotational setting, the definition of pattern is natural and easier to manipulate. We will soon make explicit when the operational and the denotational views coincide: in the next section, we show the relationship between patterns and rules, as well as between matches and pushouts.

As a side note, we hope to unify patterns and rules in a later simplification of the formalism where, at the cost of a more complex definition of $\langle \cdot, \cdot \rangle$ (namely keeping track of which elements come from the left and which elements come from the right), there is no difference between the information contained in a pattern and in a rule.

Matches A pattern is like a rule that has already integrated its two states, but there is in general no unique way to decompose a pattern into states. However, there is a much more direct definition of when a pattern can be "found" in a larger one:

Definition (Match)**.** If $(\mathfrak{A}, K_1), (\mathfrak{B}, K_2) : \Theta$, a 0-embedding $m : \mathfrak{A} \to \mathfrak{B}$ is a *match* from $(\mathfrak{A}, K_1)$ to $(\mathfrak{B}, K_2)$ whenever

- $\mathcal{B}_{\mathfrak{B}}(m(K_1), 1) \cup [\![\Delta]\!]_{\mathfrak{B}} \subseteq \mathrm{Im}(m)$ and

- $m(K_1) \subseteq K_2$

The notation is $m : (\mathfrak{A}, K_1) \to (\mathfrak{B}, K_2)$, or $m : (\mathfrak{A}, K_1) \to \mathfrak{B}$ if $K_2 = \mathrm{dom}(\mathfrak{B})$. The set K in $(\mathfrak{A}, K)$ is called the *application condition* of $(\mathfrak{A}, K)$. ■

Matches give meaning to the application condition of a pattern. As mentioned earlier, in a graph rewrite rule, a node will sometimes be marked *free*; that is: not connected to anyone. Unlike a regular node which may gain edges through a match in a larger graph, *free* nodes must not gain edge. Similarly, an element in the application condition of a pattern cannot gain an edge; as with rule application, this is expressed with a proviso of the form $\mathcal{B}_{\mathfrak{B}}(m(K_1), 1) \subseteq \mathrm{Im}(m)$.

The condition $m(K_1) \subseteq K_2$ ensures matches compose (but we will most often refine to a pattern of the form $(\mathfrak{B}, \mathrm{dom}(\mathfrak{B}))$, thus trivialising the condition).

Finally, $[\![\Delta]\!]_{\mathfrak{B}} \subseteq \mathrm{Im}(m)$ ensures that a match cannot create additional changes. To contrast with rule application: with rules, the next state is mechanically derived in the form of a pushout, so the changes $[\![\Delta]\!]$ are induced by the definition. With patterns, the set of acceptable changes must be explicitly restricted.

Matches are also defined between states; the definition just considers matches between trivial transitions created by joining a single state with itself:

Definition (State match). If $(\mathfrak{S}, K), (\mathfrak{M}, K') : \Theta^\circ$ and $m : (\langle \mathfrak{S}, \mathfrak{S} \rangle, K) \to (\langle \mathfrak{M}, \mathfrak{M} \rangle, K')$ then m *refines* $(\mathfrak{S}, K)$ to $(\mathfrak{M}, K')$ (written $m : (\mathfrak{S}, K) \to (\mathfrak{M}, K')$). ■

One maybe helpful intuition for rules and patterns is: rules & pushouts define "building a next state from a state and a transition motif" while patterns & matches define "adding context to a transition motif".

Example. The identity $\mathrm{id}_{\mathfrak{A}}$ figure 2.8 is a match from the pattern $(\mathfrak{A}, \{c\})$ to $\mathfrak{B}$. The element c is marked *free* in the precondition of $\mathfrak{A}$ to indicate that c is part of the application condition. One can check that $[\![\Delta]\!]_{\mathfrak{A}} = \{a\} \subseteq \mathrm{Im}(m)$ and that c does not gain any new link. However, b is not in the free set and the match adds a link between d and b. ■

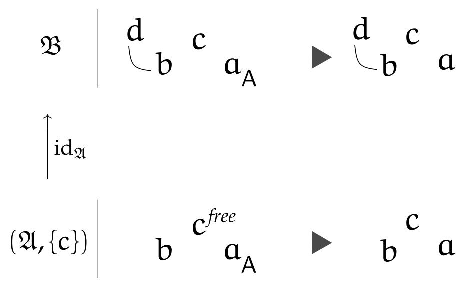


Figure 2.8.: $\mathrm{id}_{\mathfrak{A}} : (\mathfrak{A}, \{c\}) \to \mathfrak{B}$

Instances Finally it is useful to directly talk about all the transitions into which a pattern can match.

Definition (Instances). If there exists an m such that $m : (\mathfrak{A}, K) \to \mathfrak{B}$, then $\mathfrak{B}$ is an *instance* of $(\mathfrak{A}, K)$. $\mathrm{Insts}((\mathfrak{A}, K))$ is the set of instances of $(\mathfrak{A}, K)$. If P is a set of patterns, $\mathrm{Insts}(P)$ is the union of $\mathrm{Insts}((\mathfrak{A}, K))$ for $(\mathfrak{A}, K) \in P$. ■

Note that instances only consider transitions, not other patterns. In the match of figure 2.8, $\mathfrak{B}$ is an instance of $(\mathfrak{A}, \{c\})$.

2.4.1. Properties of matches

In this subsection we give useful properties of matches. The first is that an isomorphism is always a match:

Lemma 2.4.1. If $m : \mathfrak{A} \to \mathfrak{B}$ is an isomorphism, then $m : (\mathfrak{A}, K_A) \to (\mathfrak{B}, K_B)$ whenever $m(K_A) \subseteq K_B$.

Proof. $m(\mathrm{dom}(\mathfrak{A})) = \mathrm{dom}(\mathfrak{B})$. $\square$

The next lemma shows the effect of application conditions. Recall that with a 0-embedding, an element can not gain new 0-neighbors. With a match from a pattern $(\mathfrak{A}, K)$, elements outside of K are allowed to gain new 1-neighbors, but elements in K are *not*:

Lemma 2.4.2. If $m : (\mathfrak{A}, K) \to \mathfrak{B}$, $a \in m(K), b \in \mathrm{dom}(\mathfrak{B})$ and $\delta_{\mathfrak{B}}(a, b) = 1$ then $b \in \mathrm{Im}(m)$ and $\delta_{\mathfrak{A}}(m^{-1}(a), m^{-1}(b)) = 1$.

Proof. By definition of a match $b \in \mathrm{Im}(m)$. There are c, d such that $\delta_{\mathfrak{B}}(a, c) = 0$, (c, d) or $(d, c) \in [\![\mathsf{Link}]\!]_{\mathfrak{B}} \cup [\![\mathsf{Link}^\star]\!]_{\mathfrak{B}}$ and $\delta_{\mathfrak{B}}(d, b) = 0$. Applying lemma 2.2.2 yields $c, d \in \mathrm{Im}(m)$ and $\delta_{\mathfrak{A}}(m^{-1}(a), m^{-1}(c)) = 0$ and $\delta_{\mathfrak{A}}(m^{-1}(b), m^{-1}(d)) = 0$. Since m is an embedding, $\delta_{\mathfrak{A}}(m^{-1}(c), m^{-1}(d)) \leqslant 1$, so $\delta_{\mathfrak{A}}(m^{-1}(a), m^{-1}(b)) \leqslant 1$. $\delta_{\mathfrak{B}}(a, b) \leqslant \delta_{\mathfrak{A}}(m^{-1}(a), m^{-1}(b))$ so $\delta_{\mathfrak{A}}(a, b) = 1$. $\square$

Just as the "no new 0-neighbor" property helps show that 0-embeddings compose, the previous lemma helps show that matches compose:

Lemma 2.4.3. If $m_1 : (\mathfrak{A}_1, K_1) \to (\mathfrak{A}_2, K_2)$ and $m_2 : (\mathfrak{A}_2, K_2) \to (\mathfrak{A}_3, K_3)$, then $m_2 \circ m_1 : (\mathfrak{A}_1, K_1) \to (\mathfrak{A}_3, K_3)$.

Proof. The nontrivial part is showing $\mathcal{B}_{\mathfrak{A}_3}(m_2 \circ m_1(K_1), 1) \subseteq m_2 \circ m_1(\mathfrak{A}_1)$. Consider $a \in m_2(m_1(K_1))$ and b such that $\delta_{\mathfrak{A}_3}(a, b) \leqslant 1$. Since $m_1(K_1) \subseteq K_2$ and $\mathcal{B}_{\mathfrak{A}_3}(m_2(K_2), 1) \subseteq m_2(\mathfrak{A}_2)$, $b \in m_2(\mathfrak{A}_2)$. By lemmas 2.2.2 or 2.4.2 (depending on whether $\delta_{\mathfrak{A}_3}(a, b) = 0$ or 1), $\delta_{\mathfrak{A}_2}(m_2^{-1}(a), m_2^{-1}(b)) \leqslant 1$. Since $m_2^{-1}(a) \in m_1(K_1)$ and $\mathcal{B}_{\mathfrak{A}_2}(m_1(K_1), 1) \subseteq m_1(\mathfrak{A}_1)$, $m_2^{-1}(b) \in m_1(\mathfrak{A}_1)$.
The rest is by lemma 2.2.3. $\square$

The following lemma is a form of weakening: if a match guarantees an application condition K_A and relies on an application condition K_B, then it guarantees any condition K'_A weaker than K_A and it relies on any condition K'_B stronger than K_B:

Lemma 2.4.4. If $m : (\mathfrak{A}, K_A) \to (\mathfrak{B}, K_B)$, $K'_A \subseteq K_A$, and $K_B \subseteq K'_B$, then $m : (\mathfrak{A}, K'_A) \to (\mathfrak{B}, K'_B)$.

Proof. Apply lemma 2.4.1 twice on the identity, and lemma 2.4.3 once.
$\square$

The next lemma is important. It generalises the "no new 1-neighbor" consequence of application conditions: if an element is at distance d of the application condition boundary (on the inside), then it has no new $(d+1)$-neighbor:

Lemma 2.4.5. If $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}(K, d)) \to \mathfrak{B}$, $a \in m(K), b \in \mathrm{dom}(\mathfrak{B})$ and $\delta_{\mathfrak{B}}(a, b) \leqslant d + 1$ then $b \in m(\mathfrak{A})$ and $\delta_{\mathfrak{A}}(m^{-1}(a), m^{-1}(b)) = \delta_{\mathfrak{B}}(a, b)$.

Proof. By induction on $k := \delta_{\mathfrak{B}}(a, b)$. If 0, lemma 2.2.2. Otherwise true for $k' < k$ and there is $c \in \mathrm{dom}(\mathfrak{B})$ such that $\delta_{\mathfrak{B}}(a, c) = k - 1$ and $\delta_{\mathfrak{B}}(b, c) = 1$. By induction $\delta_{\mathfrak{A}}(m^{-1}(a), m^{-1}(c)) = k - 1 \leqslant d$, so $c \in m(\mathcal{B}_{\mathfrak{A}}(K, d))$, so by lemma 2.4.2: $b \in m(\mathfrak{A})$ and $\delta_{\mathfrak{A}}(m^{-1}(c), m^{-1}(b)) = 1$; so $\delta_{\mathfrak{A}}(m^{-1}(a), m^{-1}(b)) = k$ (distances do not get longer in an embedding). $\square$

Now that the proper notions are introduced, we can connect patterns and matches on the one hand, with rules and pushouts on the other.

2.5. Patterns and rules

In this section, we show that we can go back and forth between rule application and pattern match if the pattern is pure and the transition respects a technical condition.

We start with simple technical definitions and introduce a partial converse of joining states together: a way to split a transition into a precondition state and a postcondition state. We also introduce the notion of support, which is central in this thesis. Recall the distinguished predicates Has and $\mathsf{Has}^\star$ from section 2.2: a transition is supported if for all its elements, at least of one Has and $\mathsf{Has}^\star$ is true.

We start with some general notions necessary for the sequel. The symmetry of transition signatures can be used to flip a transition around:

Definition (Mirror of a transition)**.** If $\mathfrak{A} : \Theta$, its *mirror* $\mathfrak{A}^{-1}$ is $\mathfrak{A}$ with the interpretations of **Dyn** and $\mathbf{Dyn}^\star$ flipped; that is, $[\![\mathsf{A}]\!]_{\mathfrak{A}^{-1}} = [\![\mathsf{A}^\star]\!]_{\mathfrak{A}}$ and $[\![\mathsf{A}^\star]\!]_{\mathfrak{A}^{-1}} = [\![\mathsf{A}]\!]_{\mathfrak{A}}$ for every $\mathsf{A} \in \mathbf{Dyn}$. ■

The next definition singles out the dynamic unary predicate Has. There is no unique way to decompose a transition into states, so given a transition, we pick the functional closure of the interpretation of Has as the substructure representing the "before" state, and the functional closure of the interpretation of $\mathsf{Has}^\star$ as the substructure representing the "after" state.

Definition (Pre and post states)**.** If $\mathfrak{A} : \Theta$, $\mathsf{Pre}(\mathfrak{A}) : \Theta^\circ$ is the reduct of $\mathfrak{A} \restriction [\![\mathsf{Has}]\!]_{\mathfrak{A}}$ to Θ° and $\mathsf{Post}(\mathfrak{A})$ is $\mathsf{Pre}(\mathfrak{A}^{-1})$. ■

Note that in general, it is *not* the case that $\mathsf{Pre}(\langle \mathfrak{S}, \mathfrak{S}' \rangle) = \mathfrak{S}$ and $\mathsf{Post}(\langle \mathfrak{S}, \mathfrak{S}' \rangle) = \mathfrak{S}'$.

The next definition is used throughout this thesis:

Definition (Supported transition)**.** If $\mathfrak{A} : \Theta$, $\mathfrak{A}$ is *supported* whenever $\mathrm{dom}(\mathfrak{A}) = [\![\mathsf{Has}]\!]_{\mathfrak{A}} \cup [\![\mathsf{Has}^\star]\!]_{\mathfrak{A}}$. ■

Definition (Strongly supported transition)**.** If $\mathfrak{A} : \Theta$, $\mathfrak{A}$ is *strongly supported* whenever for all $a \in \mathrm{dom}(\mathfrak{A})$, $\mathcal{B}_{\mathfrak{A}}(a, 0) \subseteq [\![\mathsf{Has}]\!]_{\mathfrak{A}}$ or $\mathcal{B}_{\mathfrak{A}}(a, 0) \subseteq [\![\mathsf{Has}^\star]\!]_{\mathfrak{A}}$. ■

Any strongly supported transition is of course supported. Support means that a transition can be split into two states without information loss. Strong support means that those two states can be merged back together using $\langle \cdot, \cdot \rangle$.

Lemma 2.5.1. If $\mathfrak{A}$ is strongly supported then $\langle \mathsf{Pre}(\mathfrak{A}), \mathsf{Post}(\mathfrak{A}) \rangle = \mathfrak{A}$

Proof. Trivial. □

The next lemma goes *from pushout to match*: given the application of some rule $(\mathfrak{S}, K, \mathfrak{S}')$ to a state $\mathfrak{M}$, producing a new state $\mathfrak{M}'$, when can one build a match from $\langle \mathfrak{S}, \mathfrak{S}' \rangle$ to $\langle \mathfrak{M}, \mathfrak{M}' \rangle$?

Lemma 2.5.2. If $\mathfrak{M} \rightarrow_{(\mathfrak{S},K,\mathfrak{S}')} \mathfrak{M}'$ with pushout witnesses f, g, then for any $K' \subseteq \mathrm{dom}(\mathfrak{S}') \setminus \mathrm{dom}(\mathfrak{S})$, $(f \uplus g) : (\langle \mathfrak{S}, \mathfrak{S}' \rangle, K \uplus K') \rightarrow \langle \mathfrak{M}, \mathfrak{M}' \rangle$.

Proof.

***f* ⊎ *g* is a $\mathcal{O}$-embedding** $f \uplus g$ is injective. Now we show the structure-preserving properties. First, for all $A \in$ **Dyn**:

$$\begin{aligned}
&[\![A]\!]_{\langle \mathfrak{M},\mathfrak{M}'\rangle} \restriction \mathrm{Im}(f \uplus g) \\
&= [\![A]\!]_{\mathfrak{M}} \restriction \mathrm{Im}(f \uplus g) && \text{by def. of } \langle \mathfrak{M},\mathfrak{M}'\rangle \\
&= [\![A]\!]_{\mathfrak{M}} \restriction \mathrm{Im}(f) && \text{by } \mathrm{dom}(\mathfrak{M}) \cap \mathrm{Im}(g) = \emptyset \\
&= f([\![A]\!]_{\mathfrak{G}}) && \text{since } f \text{ is an embedding} \\
&= (f \uplus g)([\![A]\!]_{\mathfrak{G}}) && g \text{ defined nowhere on } \mathrm{dom}(\mathfrak{G}) \\
&= (f \uplus g)([\![A]\!]_{\langle \mathfrak{G},\mathfrak{G}'\rangle}) && \text{by def. of } \langle \mathfrak{G},\mathfrak{G}'\rangle
\end{aligned}$$

Now for all $A^\star \in$ **Dyn***:

$$\begin{aligned}
&[\![A^\star]\!]_{\langle \mathfrak{M},\mathfrak{M}'\rangle} \restriction \mathrm{Im}(f \uplus g) \\
&= [\![A]\!]_{\mathfrak{M}'} \restriction \mathrm{Im}(f \uplus g) && \text{by def. of } \langle \mathfrak{M},\mathfrak{M}'\rangle \\
&= \big([\![A]\!]_{\mathfrak{M}} \restriction \mathrm{Im}(f \uplus g)\big) \setminus f\big([\![A]\!]_{\mathfrak{G}} \setminus [\![A]\!]_{\mathfrak{G}'}\big) \cup (f \uplus g)\big([\![A]\!]_{\mathfrak{G}'} \setminus [\![A]\!]_{\mathfrak{G}}\big) && \text{by def. of } [\![A]\!]_{\mathfrak{M}'} \\
&= f\big([\![A]\!]_{\mathfrak{G}}\big) \setminus f\big([\![A]\!]_{\mathfrak{G}} \setminus [\![A]\!]_{\mathfrak{G}'}\big) \cup (f \uplus g)\big([\![A]\!]_{\mathfrak{G}'} \setminus [\![A]\!]_{\mathfrak{G}}\big) && \text{by assumption on } f, g \\
&= (f \uplus g)\big([\![A]\!]_{\mathfrak{G}} \setminus ([\![A]\!]_{\mathfrak{G}} \setminus [\![A]\!]_{\mathfrak{G}'}) \cup ([\![A]\!]_{\mathfrak{G}'} \setminus [\![A]\!]_{\mathfrak{G}})\big) && \text{by injectivity of } f \uplus g \\
&= (f \uplus g)\big([\![A]\!]_{\mathfrak{G}'}\big) && \text{trivial set operations} \\
&= (f \uplus g)\big([\![A^\star]\!]_{\langle \mathfrak{G},\mathfrak{G}'\rangle}\big) && \text{by def. of } \langle \mathfrak{G},\mathfrak{G}'\rangle
\end{aligned}$$

For all $T \in$ **Stat**, by definition of $f \uplus g$ and the fact that f is an embedding :

$$[\![T]\!]_{\mathfrak{M}} \restriction \mathrm{Im}(f \uplus g) = [\![T]\!]_{\mathfrak{M}} \restriction \mathrm{Im}(f) = f([\![T]\!]_{\mathfrak{G}}) = (f \uplus g)([\![T]\!]_{\mathfrak{G}}) \tag{2.3}$$

Moreover, by definition of $\mathrm{dom}(\mathfrak{M}')$:

$$f(\mathrm{dom}(\mathfrak{G}) \setminus \mathrm{dom}(\mathfrak{G}')) \cap \mathrm{dom}(\mathfrak{M}') = \emptyset \tag{2.4}$$

So:

$$\begin{aligned}
&[\![T]\!]_{\mathfrak{M}'} \restriction \mathrm{Im}(f \uplus g) \\
&= [\![T]\!]_{\mathfrak{M}} \restriction \mathrm{dom}(\mathfrak{M}') \restriction \mathrm{Im}(f \uplus g) \cup (f \uplus g)([\![T]\!]_{\mathfrak{G}'}) && \text{by def. of } [\![T]\!]_{\mathfrak{M}'} \\
&= [\![T]\!]_{\mathfrak{M}} \restriction \mathrm{dom}(\mathfrak{M}') \restriction \mathrm{Im}(f) \cup (f \uplus g)([\![T]\!]_{\mathfrak{G}'}) && \text{since } \mathrm{Im}(g) \cap \mathrm{dom}(\mathfrak{M}') = \emptyset \\
&= f([\![T]\!]_{\mathfrak{G}}) \restriction \mathrm{dom}(\mathfrak{M}') \cup (f \uplus g)([\![T]\!]_{\mathfrak{G}'}) && \text{since } f \text{ is an embedding} \\
&= f([\![T]\!]_{\mathfrak{G}} \restriction \mathrm{dom}(\mathfrak{G}')) \cup (f \uplus g)([\![T]\!]_{\mathfrak{G}'}) && \text{by (2.4)} \\
&= f([\![T]\!]_{\mathfrak{G}'} \restriction \mathrm{dom}(\mathfrak{G})) \cup (f \uplus g)([\![T]\!]_{\mathfrak{G}'}) && \text{since } \langle \mathfrak{G},\mathfrak{G}'\rangle \text{ is defined} \\
&= (f \uplus g)([\![T]\!]_{\mathfrak{G}'}) && \text{trivial} \qquad (2.5)
\end{aligned}$$

Therefore :

$$\begin{aligned}
&[\![T]\!]_{\langle \mathfrak{M},\mathfrak{M}'\rangle} \restriction \mathrm{Im}(f \uplus g) \\
&= \big([\![T]\!]_{\mathfrak{M}} \restriction \mathrm{Im}(f \uplus g)\big) \cup \big([\![T]\!]_{\mathfrak{M}'} \restriction \mathrm{Im}(f \uplus g)\big) && \text{by def. of } \langle \mathfrak{M},\mathfrak{M}'\rangle \\
&= (f \uplus g)\big([\![T]\!]_{\mathfrak{G}}\big) \cup (f \uplus g)\big([\![T]\!]_{\mathfrak{G}'}\big) && \text{by (2.3) (left), (2.5) (right)} \\
&= (f \uplus g)\big([\![T]\!]_{\langle \mathfrak{G},\mathfrak{G}'\rangle}\big) && \text{by def. of } \langle \mathfrak{G},\mathfrak{G}'\rangle
\end{aligned}$$

Now we know that $(f \uplus g)$ is an embedding. To show that it is a 0-embedding, we show that if $(b, a) \in [\![h]\!]_{\langle\mathfrak{M},\mathfrak{M}'\rangle}$ and $a \in \mathrm{Im}(f \uplus g)$, then $b \in \mathrm{Im}(f \uplus g)$ (the other direction is obvious). The rest is by induction on the minimal number of 0-edges between a and b in $G_{\langle\mathfrak{M},\mathfrak{M}'\rangle}$.

If $a \in \mathrm{Im}(f)$, suppose $b \in \mathrm{dom}(\mathfrak{M})$. Then $b \in \mathrm{Im}(f)$ since f is a 0-embedding. If $b \notin \mathrm{dom}(\mathfrak{M})$, then by definition of $\mathrm{dom}(\mathfrak{M}')$, $b \in \mathrm{Im}(g)$.

If $a \in \mathrm{Im}(g)$, then $(f \uplus g)^{-1}(a) \notin \mathrm{dom}(\mathfrak{M})$, so $(b, a) \in (f \uplus g)([\![h]\!]_{\mathfrak{G}'})$, so $b \in \mathrm{Im}(f \uplus g)$.

$[\![\boldsymbol{\Delta}]\!]_{\langle\mathfrak{M},\mathfrak{M}'\rangle} \subseteq \mathbf{Im}(\boldsymbol{f} \uplus \boldsymbol{g})$ For all $A \in$ Dyn, by definition of $\mathfrak{M}, \mathfrak{M}'$, $\boldsymbol{a} \in [\![A]\!]_{\mathfrak{M}} \setminus [\![A]\!]_{\mathfrak{M}'}$ implies $\boldsymbol{a} \in \mathrm{Im}(f)$ and $\boldsymbol{a} \in [\![A]\!]_{\mathfrak{M}'} \setminus [\![A]\!]_{\mathfrak{M}}$ implies $(f \uplus g)([\![A]\!]_{\mathfrak{G}'} \setminus [\![A]\!]_{\mathfrak{G}})$.

$\mathcal{B}_{\langle\mathfrak{M},\mathfrak{M}'\rangle}((\boldsymbol{f} \uplus \boldsymbol{g})(\boldsymbol{K} \uplus \boldsymbol{K}'), \mathbf{1}) \subseteq \mathbf{Im}(\boldsymbol{f} \uplus \boldsymbol{g})$ Let $e \in K \uplus K'$, $d \in \mathrm{dom}(\langle\mathfrak{M},\mathfrak{M}'\rangle)$, $a := (f \uplus g)(e)$ and $\delta_{\langle\mathfrak{M},\mathfrak{M}'\rangle}(a', b) \leqslant 1$.

By definition there are b, c such that $\delta_{\langle\mathfrak{M},\mathfrak{M}'\rangle}(a, b) = 0$, (b, c) or $(c, b) \in [\![\mathsf{Link}]\!]_{\langle\mathfrak{M},\mathfrak{M}'\rangle} \cup [\![\mathsf{Link}^\star]\!]_{\langle\mathfrak{M},\mathfrak{M}'\rangle}$, and $\delta_{\langle\mathfrak{M},\mathfrak{M}'\rangle}(c, d) = 0$. Since $f \uplus g$ is a 0-embedding, we have: $c \in \mathrm{Im}(f \uplus g)$ implies $d \in \mathrm{Im}(f \uplus g)$

So it now suffices to show $c \in \mathrm{Im}(f \uplus g)$.

- If the link is in $[\![\mathsf{Link}^\star]\!]_{\langle\mathfrak{M},\mathfrak{M}'\rangle} \setminus [\![\mathsf{Link}]\!]_{\langle\mathfrak{M},\mathfrak{M}'\rangle}$ then by definition of $\langle\mathfrak{M},\mathfrak{M}'\rangle$, $c \in \mathrm{Im}(f \uplus g)$.
- Otherwise the link is in $[\![\mathsf{Link}]\!]_{\langle\mathfrak{M},\mathfrak{M}'\rangle} = [\![\mathsf{Link}]\!]_{\mathfrak{M}}$, so $b, c \in \mathrm{dom}(\mathfrak{M})$. Now suppose $a \in \mathrm{dom}(\mathfrak{M})$ and $\delta_{\mathfrak{M}}(a, b) = 0$. Therefore $e \in K$ (since $(f \uplus g)(K')$ does not touch $\mathrm{dom}(\mathfrak{M})$) and $\delta_{\mathfrak{M}}(a, c) \leqslant 1$, so by definition of a match $c \in \mathrm{Im}(f) \subseteq \mathrm{Im}(f \uplus g)$.

  So we will show that $a \in \mathrm{dom}(\mathfrak{M})$ and $\delta_{\mathfrak{M}}(a, b) = 0$. We know $b \in \mathrm{dom}(\mathfrak{M})$ and $\delta_{\langle\mathfrak{M},\mathfrak{M}'\rangle}(a, b) = 0$ so by lemma 2.2.1, $a \in \mathrm{dom}(\mathfrak{M})$ and $\delta_{\mathfrak{M}}(a, b) = 0$.

□

The next lemma complements the previous one by going *from match to pushout*: given a match from a pattern to a transition, when can one derive a rule from the pattern that can be applied to construct exactly that transition?

Lemma 2.5.3. If $(\mathfrak{A}, K)$ is a pure pattern, $m : (\mathfrak{A}, K) \to \langle\mathfrak{M},\mathfrak{M}'\rangle$, $[\![\mathsf{Has}]\!]_{\mathfrak{M}} = \mathrm{dom}(\mathfrak{M})$, $[\![\mathsf{Has}]\!]_{\mathfrak{M}'} = \mathrm{dom}(\mathfrak{M}')$ and $\mathcal{r} := (\mathrm{Pre}(\mathfrak{A}), K \restriction \mathrm{dom}(\mathrm{Pre}(\mathfrak{A})), \mathrm{Post}(\mathfrak{A}))$ then $\mathfrak{M} \rightarrow_{\mathcal{r}} \mathfrak{M}'$.

Proof. Let $\mathfrak{G}$ be the reduct of $\mathfrak{A} \restriction m^{-1}(\mathrm{dom}(\mathfrak{M}))$ to Θ°. Let $\mathfrak{G}'$ be the reduct of $\mathfrak{A} \restriction m^{-1}(\mathrm{dom}(\mathfrak{M}'))$ to Θ°.

First, we show $\mathfrak{G} = \mathrm{Pre}(\mathfrak{A})$. Since $\mathfrak{G}$ and $\mathrm{Pre}(\mathfrak{A})$ are both Θ°-reducts of induced substructures of $\mathfrak{A}$, $\mathrm{dom}(\mathfrak{G}) = \mathrm{dom}(\mathrm{Pre}(\mathfrak{A}))$ implies $\mathfrak{G} = \mathrm{Pre}(\mathfrak{A})$.

- If $a \in \mathfrak{G}$ then $m(a) \in \mathrm{dom}(\mathfrak{M})$ so $m(a) \in [\![\mathsf{Has}]\!]_{\mathfrak{M}}$ by assumption on $[\![\mathsf{Has}]\!]_{\mathfrak{M}}$. Since m is an embedding, $a \in \mathrm{Pre}(\mathfrak{A})$.
- If $a \in \mathrm{Pre}(\mathfrak{A})$ then $a \in \mathfrak{A} \restriction \{b\}$ for some $b \in [\![\mathsf{Has}]\!]_{\mathfrak{A}}$; by assumption on $\mathfrak{M}, \mathfrak{M}'$, $m(b) \in \mathrm{dom}(\mathfrak{M})$. Since m is an embedding, $m(a) \in \langle\mathfrak{M},\mathfrak{M}'\rangle \restriction \{m(b)\}$, so $\delta_{\langle\mathfrak{M},\mathfrak{M}'\rangle}(a, b) = 0$. So by lemma 2.2.1, $m(a) \in \mathrm{dom}(\mathfrak{M})$ as well.

By the same reasoning $\mathfrak{G}' = \mathrm{Post}(\mathfrak{A})$. So $\mathcal{r} = (\mathfrak{G}, K \restriction \mathrm{dom}(\mathfrak{G}), \mathfrak{G}')$.

We now show $\mathfrak{M} \rightarrow_{\mathcal{r}'} \mathfrak{M}'$. Note that by lemma 2.5.1, $\langle\mathfrak{G},\mathfrak{G}'\rangle$ is defined and equal to $\mathfrak{A}$.

Since m is an embedding and $\mathfrak{M}$ is a substructure of $\langle\mathfrak{M},\mathfrak{M}'\rangle$, $\mathrm{dom}(\mathfrak{G}) = m^{-1}(\mathrm{dom}(\mathfrak{M}))$.

Let $f := m \restriction \mathrm{dom}(\mathfrak{G})$. $f : \mathfrak{G} \to \mathfrak{M}$ is an embedding, by restriction of m. To show that $\mathcal{B}_{\mathfrak{M}}(f(\mathrm{dom}(\mathfrak{G})), 0) \subseteq \mathrm{Im}(f)$, it suffices to show that for every function symbol $h \in$ **Stat**, if

$b \in \mathfrak{A} \restriction m^{-1}(\mathfrak{M})$, $a \in \mathrm{dom}(\mathfrak{M})$ and $(a,b) \in [\![h]\!]_{\mathfrak{M}}$ then $a \in \mathrm{Im}(m)$. Since $\Theta^\circ \subseteq \Theta$ and by definition of $\langle \mathfrak{M}, \mathfrak{M}' \rangle$ we have $\delta_{\langle \mathfrak{M}, \mathfrak{M}' \rangle}(a,b) = 0$ and m a 0-embedding, so by definition of a 0-embedding, $a \in \mathrm{Im}(m)$.

To get $\mathcal{B}_{\mathfrak{M}}(\mathrm{dom}(\mathfrak{G}) \setminus \mathrm{dom}(\mathfrak{G}'), 1) \subseteq \mathrm{Im}(f)$, by the above it suffices to show that if $(a,b) \in [\![\mathsf{Link}]\!]_{\mathfrak{M}}$ and $f^{-1}(a)$ is defined and in $\mathrm{dom}(\mathfrak{G}) \setminus \mathrm{dom}(\mathfrak{G}')$, then $b \in \mathrm{Im}(f)$. Take such an a. Since m is an embedding, $a \in [\![\mathsf{Has}]\!]_{\mathfrak{G}}$ and $a \notin [\![\mathsf{Has}]\!]_{\mathfrak{G}'}$, so $a \in [\![\Delta]\!]_{\mathfrak{A}}$. Since $(\mathfrak{A}, K)$ is pure, $[\![\Delta]\!]_{\mathfrak{A}} \subseteq K$, so $b \in \mathrm{Im}(f)$.

We obtain the rule application condition $\mathcal{B}_{\mathfrak{M}}(f(K \restriction \mathrm{dom}(\mathfrak{G})), 1) \subseteq \mathrm{Im}(f)$ from $\mathcal{B}_{\langle \mathfrak{M}, \mathfrak{M}' \rangle}(m(K), 1) \subseteq \mathrm{Im}(m)$ since $\mathfrak{M}$ is a substructure of $\langle \mathfrak{M}, \mathfrak{M}' \rangle$ and $\mathrm{Im}(m) \setminus \mathrm{Im}(f) \subseteq \mathrm{dom}(\mathfrak{M}') \setminus \mathrm{dom}(\mathfrak{M})$. So $f : (\mathfrak{G}, K \restriction \mathrm{dom}(\mathfrak{G})) \to \mathfrak{M}$.

Again, since m is an embedding and $\mathfrak{M}'$ is a substructure of $\langle \mathfrak{M}, \mathfrak{M}' \rangle$, $\mathrm{dom}(\mathfrak{G}') = m^{-1}(\mathrm{dom}(\mathfrak{M}'))$. By definition of $\mathfrak{G}, \mathfrak{G}'$, $m(\mathrm{dom}(\mathfrak{G}') \setminus \mathrm{dom}(\mathfrak{G})) \cap \mathrm{dom}(\mathfrak{M}) = \emptyset$; so let $g : \mathrm{dom}(\mathfrak{G}') \setminus \mathrm{dom}(\mathfrak{G}) \to \mathcal{U} \setminus \mathrm{dom}(\mathfrak{M}) := x \mapsto m(x)$. As a restriction of m, g is injective. Trivially, $m = f \uplus g$.

Similarly, $m(\mathrm{dom}(\mathfrak{G}) \setminus \mathrm{dom}(\mathfrak{G}')) \cap \mathrm{dom}(\mathfrak{M}') = \emptyset$. Moreover, since m is a match and $\langle \mathfrak{M}, \mathfrak{M}' \rangle$ is supported (by assumption on $\mathrm{dom}(\mathfrak{M})$ and $\mathrm{dom}(\mathfrak{M}')$), $\mathrm{dom}(\mathfrak{M}') \setminus \mathrm{dom}(\mathfrak{M}) = \mathrm{Im}(g)$ and $\mathrm{dom}(\mathfrak{M}) \setminus \mathrm{dom}(\mathfrak{M}') = f(\mathrm{dom}(\mathfrak{G}) \setminus \mathrm{dom}(\mathfrak{G}'))$. Put together :

$$\mathrm{dom}(\mathfrak{M}') = \mathrm{dom}(\mathfrak{M}) \setminus f(\mathrm{dom}(\mathfrak{G}) \setminus \mathrm{dom}(\mathfrak{G}')) \uplus \mathrm{Im}(g) \tag{2.6}$$

(note that f is defined and equal to m on $\mathrm{dom}(\mathfrak{G}) \setminus \mathrm{dom}(\mathfrak{G}')$, and that g does not map to $\mathrm{dom}(\mathfrak{M})$).

For all $\mathsf{A} \in \mathbf{Dyn}$:

$$\begin{aligned}
[\![\mathsf{A}]\!]_{\mathfrak{M}'} &= [\![\mathsf{A}]\!]_{\mathfrak{M}} \setminus ([\![\mathsf{A}]\!]_{\mathfrak{M}} \setminus [\![\mathsf{A}]\!]_{\mathfrak{M}'}) \uplus ([\![\mathsf{A}]\!]_{\mathfrak{M}'} \setminus [\![\mathsf{A}]\!]_{\mathfrak{M}}) & \\
&= [\![\mathsf{A}]\!]_{\mathfrak{M}} \setminus m([\![\mathsf{A}]\!]_{\mathfrak{G}} \setminus [\![\mathsf{A}]\!]_{\mathfrak{G}'}) \uplus m([\![\mathsf{A}]\!]_{\mathfrak{G}'} \setminus [\![\mathsf{A}]\!]_{\mathfrak{G}}) & \text{since } m : (\mathfrak{A}, K) \to \langle \mathfrak{M}, \mathfrak{M}' \rangle \\
&= [\![\mathsf{A}]\!]_{\mathfrak{M}} \setminus f([\![\mathsf{A}]\!]_{\mathfrak{G}} \setminus [\![\mathsf{A}]\!]_{\mathfrak{G}'}) \uplus m([\![\mathsf{A}]\!]_{\mathfrak{G}'} \setminus [\![\mathsf{A}]\!]_{\mathfrak{G}}) & \text{by definition of } f
\end{aligned}$$

The decomposition of $\mathrm{dom}(\mathfrak{M}')$ given by (2.6) yields:

$$[\![\mathsf{T}]\!]_{\mathfrak{M}'} = [\![\mathsf{T}]\!]_{\mathfrak{M}} \restriction \mathrm{dom}(\mathfrak{M}') \cup m([\![\mathsf{T}]\!]_{\mathfrak{G}'})$$

By $[\![\mathbf{Stat}]\!]_{\mathfrak{M}} \restriction \mathcal{B}_{\mathfrak{M}}(\mathrm{dom}(\mathfrak{M}'), 0) = [\![\mathbf{Stat}]\!]_{\mathfrak{M}'} \restriction \mathcal{B}_{\mathfrak{M}'}(\mathrm{dom}(\mathfrak{M}), 0)$ (by assumption on $\langle \mathfrak{M}, \mathfrak{M}' \rangle$) for the $[\![\mathsf{T}]\!]_{\mathfrak{M}} \restriction \mathrm{dom}(\mathfrak{M}')$ part, and by $g = m \restriction \mathrm{dom}(\mathfrak{G}')$ and the fact that m is an embedding for the $m([\![\mathsf{T}]\!]_{\mathfrak{G}'})$ part.

So $\mathfrak{M}'$ is a pushout of $(f, g, \mathfrak{G}, \mathfrak{G}', \mathfrak{M})$.

□

To summarise the last two lemmas:

- If a rule applied to a state produces a pushout with witnesses f, g, then $f \uplus g$ is a match from (morally) the rule to the $\langle \text{state}, \text{pushout} \rangle$ transition.

- If there is a match m between a pure pattern and $\langle \mathfrak{M}, \mathfrak{M}' \rangle$, then the rule induced by the pattern, when applied to $\mathfrak{M}$, produces $\mathfrak{M}'$ as pushout. Moreover, the proof defines the pushout witnesses f and g such that $m = f \uplus g$.

It is easy to check that under the right conditions (pure pattern, strongly supported transitions), the composition of those lemmas in any order yields the identity function. Now that the 1-step operational and denotational views are connected, we move to traces.

2.6. Runs and executions

In this section we extend the operational and denotational views to traces (as seen for rule-based models in the introduction) and we give conditions that are sufficient to make both views coincide.

Operational view When given a starting state, a *set of rules* induces a pushout-based trace we call an execution. Since rule application (by pushout) generates states that can be joined into transitions, a set of rules can now generate sequences of transitions from a given starting state:

Definition (Executions). If $\mathtt{R}$ is a set of rules, and $\mathfrak{M}_0, \mathfrak{M}_1, \ldots$ is a possibly infinite sequence such that $\mathfrak{M}_0 \twoheadrightarrow_{\mathtt{R}} \mathfrak{M}_1 \twoheadrightarrow_{\mathtt{R}} \mathfrak{M}_2 \twoheadrightarrow_{\mathtt{R}} \ldots$, then $\langle \mathfrak{M}_0, \mathfrak{M}_1 \rangle, \langle \mathfrak{M}_1, \mathfrak{M}_2 \rangle, \ldots$ is an *execution* starting from $\mathfrak{M}_0$ induced by $\mathtt{R}$ (well-defined by lemma 2.3.1). The set of all executions starting from a state $\mathfrak{M} : \Theta^\circ$ induced by $\mathtt{R}$ is $\mathtt{exec}_{\mathfrak{M}}(\mathtt{R})$ ■

Denotational view Instead of being iteratively generated, like executions, runs are a function of *sets of transitions*. They must make sure that their Pre and Post match pairwise. This denotational view paves the way for runs defined on transitions that are models of a formula.

Definition (Run). If M is a set of transitions, the *runs* $\rho(M)$ of M are the possibly infinite sequences $r := \mathfrak{A}_1, \mathfrak{A}_2, \ldots$ of supported transitions from M such that for all $\mathfrak{A}_i, \mathfrak{A}_{i+1} \in r$ we have $\mathsf{Post}(\mathfrak{A}_i) = \mathsf{Pre}(\mathfrak{A}_{i+1})$. ■

Definition (Run from initial state). If $\mathfrak{S} : \Theta^\circ$ the $\mathfrak{S}$-runs $\rho_{\mathfrak{S}}(M)$ of M are the runs of the form $\mathfrak{A}_1, \ldots$ of M such that $\mathfrak{S} = \mathsf{Pre}(\mathfrak{A}_1)$.

■

Definition (From patterns to rules). If P is a set of patterns the set of rules $\mathtt{rules}(P)$ is:

$$\mathtt{rules}(P) := \{(\mathsf{Pre}(\mathfrak{A}), K \restriction \mathsf{dom}(\mathsf{Pre}(\mathfrak{A})), \mathsf{Post}(\mathfrak{A})) \mid (\mathfrak{A}, K) \in P\}$$

■

Contrast runs and executions: both are sequences of transitions; one is described operationally and the other denotationally. The task ahead is to find out when they are the same.

The next few technical lemmas prepare the theorem at the end of this section; in particular they show that when Has consistently encodes *presence* in a state, the operational and denotational trace generation processes maintain that consistency.

Lemma 2.6.1. If $\mathfrak{M} : \Theta^\circ$ is such that $\mathsf{dom}(\mathfrak{M}) = [\![\mathsf{Has}]\!]_{\mathfrak{M}}$, $\mathfrak{A}$ is strongly supported, and $\mathfrak{M}'$ is the pushout of $(f, g, \mathsf{Pre}(\mathfrak{A}), \mathsf{Post}(\mathfrak{A}), \mathfrak{M})$ then $\mathsf{dom}(\mathfrak{M}') = [\![\mathsf{Has}]\!]_{\mathfrak{M}'}$.

Proof. Since $\mathfrak{A}$ is strongly supported, every element in $\mathsf{dom}(\mathfrak{A}) \restriction [\![\mathsf{Has}]\!]_{\mathfrak{A}}$ is in $[\![\mathsf{Has}]\!]_{\mathfrak{A}}$ and so $[\![\mathsf{Has}]\!]_{\mathsf{Pre}(\mathfrak{A})} = \mathsf{dom}(\mathsf{Pre}(\mathfrak{A}))$. The same goes for $\mathsf{Has}^\star$ and $\mathsf{Post}(\mathfrak{A})$. The rest is trivial by definition of a pushout. □

Lemma 2.6.2. If $\mathfrak{A}$ is strongly supported, $\mathfrak{B}$ is supported and $\mathfrak{B}$ is an instance of $(\mathfrak{A}, K)$ for some K, then $\mathfrak{B}$ is strongly supported.

Proof. Suppose there are b_1, b_2 such that $b_1 \in \mathsf{dom}(\mathfrak{B} \restriction \{b_2\})$, $b_1 \notin [\![\mathsf{Has}^\star]\!]_{\mathfrak{B}}$, $b_2 \in [\![\mathsf{Has}^\star]\!]_{\mathfrak{B}}$.

As $\mathfrak{B}$ is supported, $b_1 \in [\![\Delta]\!]_{\mathfrak{B}}$, so there is $a_1 \in \mathsf{dom}(\mathfrak{A})$ such that $f(a_1) = b_1$ for some embedding f of $\mathfrak{A}$ into $\mathfrak{B}$, and some $a_2 \in \mathsf{dom}(\mathfrak{A})$ such that $a_2 \in \mathsf{dom}(\mathfrak{A} \restriction \{a_1\})$ and $a_2 \in [\![\mathsf{Has}]\!]_{\mathfrak{A}}$, $f(a_2) = b_2$. By definition of an embedding, $a_1 \notin [\![\mathsf{Has}^\star]\!]_{\mathfrak{A}}$, which contradicts the definition of strong support. □

This technical lemma simply ensures that $\mathsf{Pre}(\cdot)$, $\mathsf{Post}(\cdot)$ and $\langle\cdot,\cdot\rangle$ interact well together.

Lemma 2.6.3. If $\mathfrak{M}, \mathfrak{M}', \mathfrak{M}'' : \Theta^\circ$ and $\langle\mathfrak{M}, \mathfrak{M}'\rangle, \langle\mathfrak{M}', \mathfrak{M}''\rangle$ are defined, then $\mathsf{Post}(\langle\mathfrak{M}, \mathfrak{M}'\rangle) = \mathsf{Pre}(\langle\mathfrak{M}', \mathfrak{M}''\rangle)$.

Proof. If $a \in \mathrm{dom}(\mathsf{Post}(\langle\mathfrak{M}, \mathfrak{M}'\rangle))$ there is $b \in [\![\mathsf{Has}]\!]_{\mathfrak{M}'}$ such that $a \in \mathrm{dom}(\langle\mathfrak{M}, \mathfrak{M}'\rangle \restriction [\![\mathsf{Has}]\!]_{\mathfrak{M}'})$. Since $[\![\mathbf{Stat}]\!]_{\mathfrak{M}} \restriction \mathcal{B}_{\mathfrak{M}}(\mathrm{dom}(\mathfrak{M}'), 0) = [\![\mathbf{Stat}]\!]_{\mathfrak{M}'} \restriction \mathcal{B}_{\mathfrak{M}'}(\mathrm{dom}(\mathfrak{M}), 0)$, $a \in \mathrm{dom}(\mathfrak{M}')$. Similarly, if $a \in \mathsf{Pre}(\langle\mathfrak{M}', \mathfrak{M}''\rangle)$, $a \in \mathrm{dom}(\mathfrak{M}')$. So $\mathrm{dom}(\mathsf{Post}(\langle\mathfrak{M}, \mathfrak{M}'\rangle)) = \mathrm{dom}(\mathsf{Pre}(\langle\mathfrak{M}', \mathfrak{M}''\rangle))$. The equality of $\mathbf{Dyn}$ and $\mathbf{Stat}$ is by further unrolling of definitions. □

Main theorem We now prove the main theorem of this chapter: if patterns include all changes in their application conditions and both patterns and the starting state treat Has consistently, then the operational and denotational trace semantics of patterns are equal.

The proof is by double inclusion: from (pushout-based) executions to (match-based) runs, we show using lemma 2.5.2 that if a rule $\imath$ is induced by a pattern p, then a pushout based on $\imath$ induces a match for p. From (match-based) runs to (pushout-based) executions, we show using lemma 2.5.3 that the target of a match from a pattern p can be decomposed into the ingredients of a pushout starting from a rule induced by p.

Theorem 2.6.4. If P is a set of pure and strongly supported patterns, $\mathfrak{M}_0 : \Theta^\circ$ and $[\![\mathsf{Has}]\!]_{\mathfrak{M}_0} = \mathrm{dom}(\mathfrak{M}_0)$ then $\mathtt{exec}_{\mathfrak{M}_0}(\mathtt{rules}(P)) = \rho_{\mathfrak{M}_0}(\mathsf{Insts}(P))$

Proof.

$\mathtt{exec}_{\mathfrak{M}_0}(\mathtt{rules}(\mathbf{P})) \subseteq \rho_{\mathfrak{M}_0}(\mathbf{Insts}(\mathbf{P}))$ Let $r := \langle\mathfrak{M}_0, \mathfrak{M}_1\rangle, \ldots \in \mathtt{exec}_{\mathfrak{M}_0}(\mathtt{rules}(P))$.

By lemma 2.6.1, $[\![\mathsf{Has}]\!]_{\mathfrak{M}} = \mathrm{dom}(\mathfrak{M})$ for any $\mathfrak{M}$ that is $\rightarrow_{\mathtt{rules}(P)}$-accessible from $\mathfrak{M}_0$; this immediately implies $\langle\mathfrak{M}_i, \mathfrak{M}_{i+1}\rangle$ is supported for all $\langle\mathfrak{M}_i, \mathfrak{M}_{i+1}\rangle \in r$. By lemma 2.6.3 $\mathsf{Post}(\langle\mathfrak{M}_i, \mathfrak{M}_{i+1}\rangle) = \mathsf{Pre}(\langle\mathfrak{M}_{i+1}, \mathfrak{M}_{i+2}\rangle)$ whenever $\langle\mathfrak{M}_{i+1}, \mathfrak{M}_{i+2}\rangle \in r$.

Take any $\langle\mathfrak{M}_i, \mathfrak{M}_{i+1}\rangle \in r$. By definition of $\mathtt{exec}_{\mathfrak{M}_0}(\mathtt{rules}(P))$ there are $(\mathfrak{A}, K) \in P$, $f : (\mathsf{Pre}(\mathfrak{A}), K \restriction \mathrm{dom}(\mathsf{Pre}(\mathfrak{A}))) \to \mathfrak{M}_i$, and an injection $g : \mathrm{dom}(\mathsf{Post}(\mathfrak{A})) \setminus \mathrm{dom}(\mathsf{Pre}(\mathfrak{A})) \to \mathfrak{U} \setminus \mathfrak{M}_i$ such that $\mathfrak{M}_{i+1}$ is the pushout of $(f, g, \mathsf{Pre}(\mathfrak{A}), \mathsf{Post}(\mathfrak{A}), \mathfrak{M}_i)$.

$\mathfrak{A}$ is supported so by lemma 2.5.1, $\langle\mathsf{Pre}(\mathfrak{A}), \mathsf{Post}(\mathfrak{A})\rangle = \mathfrak{A}$. Note that $K \setminus (K \restriction \mathrm{dom}(\mathsf{Pre}(\mathfrak{A}))) \subseteq \mathrm{dom}(\mathsf{Post}(\mathfrak{A})) \setminus \mathrm{dom}(\mathsf{Pre}(\mathfrak{A}))$. Let $m := f \uplus g$

By lemma 2.5.2, $m : (\mathfrak{A}, K) \to \langle\mathfrak{M}_i, \mathfrak{M}_{i+1}\rangle$. So $r \in \rho_{\mathfrak{M}_0}(\mathsf{Insts}(P))$.

$\rho_{\mathfrak{M}_0}(\mathsf{Insts}(P)) \subseteq \mathtt{exec}_{\mathfrak{M}_0}(\mathtt{rules}(P))$ Let $r := \mathfrak{B}_1, \ldots \in \rho_{\mathfrak{M}_0}(\mathsf{Insts}(P))$.

By definition of $\rho_{\mathfrak{M}_0}(\mathsf{Insts}(P))$, each $\mathfrak{B}_i$ is supported, $\mathsf{Pre}(\mathfrak{B}_1) = \mathfrak{M}_0$ and $\mathsf{Post}(\mathfrak{B}_i) = \mathsf{Pre}(\mathfrak{B}_{i+1})$ whenever $B_{i+1} \in r$; and by lemma 2.5.1 $\langle\mathsf{Pre}(\mathfrak{B}_i), \mathsf{Post}(\mathfrak{B}_i)\rangle = \mathfrak{B}_i$.

By lemma 2.6.2, each $\mathfrak{B}_i$ is strongly supported, so $[\![\mathsf{Has}]\!]_{\mathsf{Pre}(B_i)} = \mathrm{dom}(\mathsf{Pre}(\mathfrak{B}_i))$ for every $\mathfrak{B}_i \in r$, and $[\![\mathsf{Has}]\!]_{\mathsf{Post}(B_i)} = \mathrm{dom}(\mathsf{Post}(\mathfrak{B}_i))$.

By definition of $\rho_{\mathfrak{M}_0}(\mathsf{Insts}(P))$ there is some $(\mathfrak{A}, K) \in P$ and some $m : (\mathfrak{A}, K) \to \langle\mathsf{Pre}(\mathfrak{B}_i), \mathsf{Post}(\mathfrak{B}_i)\rangle$. By lemma 2.5.3, with $\imath := (\mathsf{Pre}(\mathfrak{A}), K \restriction \mathrm{dom}(\mathsf{Pre}(\mathfrak{A})), \mathsf{Post}(\mathfrak{A}))$ we get $\mathsf{Pre}(\mathfrak{B}_i) \rightarrow_{\imath} \mathsf{Post}(\mathfrak{B}_i)$; and $\imath \in \mathtt{rules}(P)$ by definition so $\mathsf{Pre}(\mathfrak{B}_i) \rightarrow_{\mathtt{rules}(P)} \mathsf{Post}(\mathfrak{B}_i)$.

So $r \in \mathtt{exec}_{\mathfrak{M}_0}(\mathtt{rules}(P))$.

□

In figure 2.9, we see the connection between the execution of the rules for a pattern set P and the runs on the instance of P. Note that $r = (\mathsf{Pre}(\mathfrak{A}), K \restriction \mathrm{dom}(\mathsf{Pre}(\mathfrak{A})), \mathsf{Post}(\mathfrak{A}))$, and similarly for r' and r''.

$$\begin{tikzcd}[column sep=small]
& r \arrow[dl] \arrow[d, dashed, dash] & & r' \arrow[dl] \arrow[d, dashed, dash] & & r'' \arrow[dl] \arrow[d, dashed, dash] & & \text{induced by} & (\mathfrak{A}, K) \arrow[dd] & (\mathfrak{A}', K') \arrow[dd] & (\mathfrak{A}'', K'') \arrow[dd] \\
& {} \arrow[l, dashed, dash, "\text{push}"'] & & {} \arrow[l, dashed, dash, "\text{push}"'] & & {} \arrow[l, dashed, dash, "\text{push}"'] & & & & & \\
\mathfrak{M}_0 \arrow[rr, -{Triangle[open]}] & & \mathfrak{M}_1 \arrow[rr, -{Triangle[open]}] & & \mathfrak{M}_2 \arrow[rr, -{Triangle[open]}] & & \mathfrak{M}_3 & & & & \\
& \langle \mathfrak{M}_0, \mathfrak{M}_1 \rangle & & \langle \mathfrak{M}_1, \mathfrak{M}_2 \rangle & & \langle \mathfrak{M}_2, \mathfrak{M}_3 \rangle & & = & \langle \mathfrak{M}_0, \mathfrak{M}_1 \rangle & \langle \mathfrak{M}_1, \mathfrak{M}_2 \rangle & \langle \mathfrak{M}_2, \mathfrak{M}_3 \rangle
\end{tikzcd}$$

Figure 2.9.: Left: an execution in $\mathtt{exec}_{\mathfrak{M}_0}(\mathtt{rules}(P))$. Right a run in $\rho_{\mathfrak{M}_0}(\mathsf{Insts}(P))$.

3. Execution of change-minimal specifications

In this chapter, we study formulas on transitions and states and further explore the connection between runs and executions.

Formulas and $\downarrow$ We start in section 3.1 by defining a change order on transitions which describes "how much" a transition does relative to another. Then in section 3.2 we define logic formulas on transitions, including the minimisation operator, which specifies the change-minimal models of a formula, as hinted at in the introduction. This defines FO[$\downarrow$], which is first-order logic extended with a unary minimisation operator. After section 3.2, we derive a notion of *run* on formulas, and the rest of the chapter is about the runs associated with formulas and finding rules which execute in a way that corresponds with the runs.

Runs on models, runs on instances First, we will introduce a parameterised function from formulas to patterns (the canonical patterns). We define those in section 3.3. For now consider that the canonical patterns provide a good "summary" of what the formula is about. Patterns can be instantiated, and the set of instances of the canonical patterns of a formula overapproximates the models of that formula.

Now there are three ways to look at runs based on ϕ:

1. We can consider the runs on the *models* of ϕ.
2. We can instantiate the canonical patterns of ϕ and consider the runs of those *instances*.
3. We can convert those canonical patterns into rules and consider the *execution* of those rules.

Thanks to the previous chapter, we know conditions where 2. and 3. coincide. The rest of the chapter after section 3.2 will be dedicated to building formulas where 1. and 2. coincide.

This will not be the end of the story. The resulting execution may require an infinite number of rules. The following chapter will address the issue.

Note on figures: in the sequel, we move to a more comfortable representation of transitions. Elements in figures belong to a single unary predicate from $\mathbf{Stat}$*, such as K or E. This predicate is shown within a large circle. Around the circle, identifers such as* $\mathfrak{a}, \mathfrak{b}, \ldots$ *are as before, and* sans-serif *letters represent belonging to a dynamic predicate of* $\mathbf{Dyn}$ *(left of the arrow) or of* $\mathbf{Dyn}^\star$ *(right of the arrow).*

3.1. Change order

Here we define an order $\trianglelefteq$ between transitions such that if $\mathfrak{A} \trianglelefteq \mathfrak{B}$, $\mathfrak{A}$ can be said to *do less than* $\mathfrak{B}$. Recall that change is the difference between the interpretation of $\mathbf{Dyn}$ and the interpretation of $\mathbf{Dyn}^\star$, so we consider the product subset relation to compare changes. With the order $\trianglelefteq$, we can then define minimisation, as we shall see in the next section.

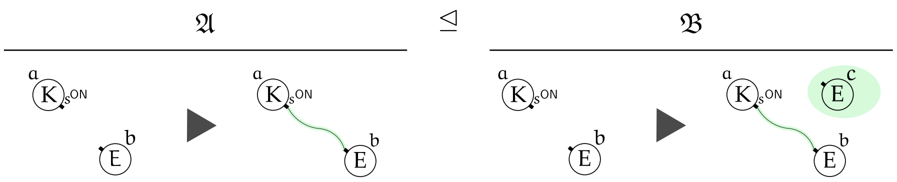


Figure 3.1.: $\mathfrak{A} \trianglelefteq \mathfrak{B}$. $\mathsf{F} = \neg\mathsf{Link}(a,b) \wedge \mathsf{Link}^\star(a,b)$

As mentioned in the introduction, specifications on transitions are too lax when it comes to having any hope of generating traces. To motivate the next section, we informally use formulas. If we write $\neg\mathsf{Link}(a,b) \wedge \mathsf{Link}^\star(a,b)$ (to require that a and b become linked), we are not explicitly forbidding additional changes; such as the creation of a brand new element c.

However, if we wish to generate traces using this formula, we should make the additional assumption that *all observations have been accounted for*. If a c could appear exactly when a and b become linked, we would know it.

Take two transitions $\mathfrak{A}, \mathfrak{B}$ with the exact same precondition but where

- $\mathfrak{A}$ creates an a–b link.
- $\mathfrak{B}$ creates a–b link *and* creates c.

We would like a way to *compare* $\mathfrak{A}$ to $\mathfrak{B}$ and conclude that while $\mathfrak{A}$ is a good model of our formula when *all observations have been accounted for*, $\mathfrak{B}$ is not. $\mathfrak{A}$ and $\mathfrak{B}$ can be seen on figure 3.1.

Definition (Change order)**.** If $\mathfrak{A}, \mathfrak{B} : \Theta$, we say that $\mathfrak{A} \trianglelefteq \mathfrak{B}$ whenever:

(1) $[\![\mathbf{Dyn}]\!]_{\mathfrak{A}} = [\![\mathbf{Dyn}]\!]_{\mathfrak{B}}$

(2) $[\![\mathbf{Stat}]\!]_{\mathfrak{A}} = [\![\mathbf{Stat}]\!]_{\mathfrak{B}} \upharpoonright \mathrm{dom}(\mathfrak{A})$

(3) $[\![\Delta\mathsf{A}]\!]_{\mathfrak{A}} \subseteq [\![\Delta\mathsf{A}]\!]_{\mathfrak{B}}$ for all $\mathsf{A} \in \mathbf{Dyn}$.

(4) $\mathrm{dom}(\mathfrak{A}) \subseteq \mathrm{dom}(\mathfrak{B})$

■

Example. In the biology-inspired example of figure 3.1, the transition $\mathfrak{B}$ has two changes: the enzyme b and the kinase a become linked, and an enzyme c is created (changes are highlighted in green). $\mathfrak{A}$ is strictly below $\mathfrak{B}$ in the change order. It has the same precondition ($[\![\mathbf{Dyn}]\!]_{\mathfrak{A}} = [\![\mathbf{Dyn}]\!]_{\mathfrak{B}}$), but c is not in $\mathrm{dom}(\mathfrak{A})$ ($\mathrm{dom}(\mathfrak{A}) \subseteq \mathrm{dom}(\mathfrak{B})$). On the *remaining* elements, static interpretations are the same ($[\![\mathbf{Stat}]\!]_{\mathfrak{A}} = [\![\mathbf{Stat}]\!]_{\mathfrak{B}} \upharpoonright \mathrm{dom}(\mathfrak{A})$). Finally it contains strictly fewer changes ($[\![\Delta\mathsf{A}]\!]_{\mathfrak{A}} \subseteq [\![\Delta\mathsf{A}]\!]_{\mathfrak{B}}$ for all $\mathsf{A} \in \mathbf{Dyn}$); indeed the a–b link still appears, but no enzyme is created. Importantly, $\mathfrak{A}$ contains no additional change.

Formally, the "creation" of c displayed at the top of figure 3.1 is a representation of $c \notin [\![\mathsf{A}]\!]_{\mathfrak{B}}$ for all $\mathsf{A} \in \mathbf{Dyn}$ and $c \in [\![\mathsf{A}^\star]\!]_{\mathfrak{B}}$ for some $\mathsf{A}^\star \in \mathbf{Dyn}^\star$, typically $\mathsf{Has}^\star$. It is possible for c to not be in $\mathrm{dom}(\mathfrak{A})$ precisely because c does not appear anywhere in the interpretation of $\mathbf{Dyn}$.

In addition, note that support enforces some degree of coherence : since every element is in the interpretation of Has or of $\mathsf{Has}^\star$, no element is a "ghost", formally present, but that could be removed by going down the change order $\trianglelefteq$ even though the strictly smaller model would have no fewer changes than the larger one.

■

Lemma 3.1.1. $\trianglelefteq$ is a partial order

Proof. Reflexivity and transitivity are trivial. For antisymmetry, assume $\mathfrak{A} \trianglelefteq \mathfrak{B}$ and $\mathfrak{B} \trianglelefteq \mathfrak{A}$. We immediately have $[\![\mathbf{Dyn}]\!]_{\mathfrak{A}} = [\![\mathbf{Dyn}]\!]_{\mathfrak{B}}$ as well as $[\![\mathbf{Stat}]\!]_{\mathfrak{A}} = [\![\mathbf{Stat}]\!]_{\mathfrak{B}}$ and $\mathrm{dom}(\mathfrak{A}) = \mathrm{dom}(\mathfrak{B})$. We just need $[\![\mathbf{Dyn}^\star]\!]_{\mathfrak{A}} = [\![\mathbf{Dyn}^\star]\!]_{\mathfrak{B}}$. For every $A \in \mathbf{Dyn}$, $[\![A^\star]\!]_{\mathfrak{A}} = [\![A^\star]\!]_{\mathfrak{B}}$ is given by $[\![\Delta A]\!]_{\mathfrak{A}} = [\![\Delta A]\!]_{\mathfrak{B}}$, $[\![A]\!]_{\mathfrak{A}} = [\![A]\!]_{\mathfrak{B}}$, and simple set-theoretic considerations. So $\mathfrak{A} = \mathfrak{B}$.

□

3.2. First-order logic on transitions, ↓

In this section we introduce FO[↓], which is first-order logic augmented with a $\trianglelefteq$-minimising unary operator, ↓. Remember that $\trianglelefteq$ orders transitions according to how much change they contain.

We start with first-order logic with equality (FO) on the signature Θ. Here are some examples of FO formulas on transitions:

Examples.

$\neg \mathsf{Has}(x) \wedge \mathsf{Has}^\star(x)$	x appears
$\exists y.\ \mathsf{Link}(x,y) \wedge (\mathsf{On}(y) \leftrightarrow \mathsf{On}(x))$	some y is linked to x, x is On iff y is
$\forall x.\ \mathsf{Off}(x) \rightarrow \forall y.\ \neg \mathsf{Link}(x,y)$	Any Off element is free

■

3.2.1. Minimisation (↓)

To talk about the change order defined earlier, we introduce a unary operator ↓. Its intended meaning is precisely : *all observations have been accounted for*. Going back to figure 3.1, note that both $\mathfrak{A}$ and $\mathfrak{B}$ are models of F. But we would like only $\mathfrak{A}$ to be a model of $\downarrow F$, not $\mathfrak{B}$.

We extend first-order logic with equality to FO[↓], where ↓ is a unary logical constant whose semantics are defined by induction on the structure of ϕ.

Definition (↓). If $\phi \in$ FO[↓], $\mathfrak{A} : \Theta$ and $\mu : \langle \phi \rangle \rightarrow \mathrm{dom}(\mathfrak{A})$:

$$\mathfrak{A}, \mu \vDash \downarrow \phi$$
$$:=$$
$$\mathfrak{A}, \mu \vDash \phi \text{ and } \mathfrak{B}, \mu \nvDash \phi \text{ for all } \mathfrak{B} \triangleleft \mathfrak{A} \text{ such that } \mathrm{Im}(\mu) \subseteq \mathrm{dom}(\mathfrak{B})$$

■

$\downarrow\phi$ is pronounced "min ϕ".

From now on (and until we explicitly move to 2nd order logic in section 7.1), "a formula" means "a Θ-formula over FO[↓]".

3.2.2. FO[↓] generalities

In this section we informally clarify logic notions. An FO[↓] formula is built from terms, atoms, and the usual $\wedge, \vee, \neg, \forall, \exists$ and ↓. The satisfaction relation is defined on non-empty structures only.

Notation (Literal sets). If Σ is a signature, $\mathcal{L}_\Sigma$ is the set of Σ-literals, and $\mathcal{L}^{=}_{\Sigma}$ the union of all equality literals and $\mathcal{L}_\Sigma$. ■

Notation (Free variables)**.** If ϑ is a term, a set of terms, or a formula, $\langle\vartheta\rangle$ is the set of free variables in ϑ.

If we write $\phi(x_1, \ldots, x_n)$, we mean that $\langle\phi\rangle \subseteq \{x_1, \ldots x_n\}$. ■

Notation (Quantifier closure)**.** If ϕ is a formula, $\exists\phi$ is the existential closure of ϕ and $\forall\phi$ is its universal closure. ■

Notation (Quantifier rank)**.** If ϕ is a formula, $\|\phi\|$ is the quantifier rank of ϕ. ■

Convention. We assume that variables are always fresh. We reuse names such as $x, y, \ldots$ etc to make reading easier, but name capture only happens through binders such as $\forall\, x$ or explicitly when variables are surfaced as arguments of formulas, such as in $\phi_1(x) \wedge \phi_2(x)$. ■

Definition (Semantic bracket)**.** We use the semantic bracket for formulas and terms, e.g. $[\![\phi]\!]$ is the set of pairs $(\mathfrak{A}, \mu)$ with $\mu : \langle\phi\rangle \to \mathrm{dom}(\mathfrak{A})$ such that $\mathfrak{A}, \mu \vDash \phi$. With $\langle\phi\rangle = x_1, \ldots x_n$ (we assume some ordering on variables), $[\![\phi]\!]_{\mathfrak{A}}$ is the set of n-tuples $(a_1, \ldots, a_n)$ of elements of $\mathrm{dom}(\mathfrak{A})$ such that $\mathfrak{A}, x_1 \mapsto a_1 \ldots x_n \mapsto a_n \vDash \phi$.

If t is a term and $\mu : \langle t\rangle \to \mathrm{dom}(\mathfrak{A})$, $[\![t]\!]_{\mathfrak{A},\mu}$ is the interpretation of t in $\mathfrak{A}, \mu$ defined by induction on the structure of t. ■

Guards

Definition (Guards)**.** If $x \neq y$, a *guard* $\alpha(x, y)$ is a binary atom (including "=") where both x and y occur. ■

Note that unlike with the notation $\phi(x, y)$, $\alpha(x, y)$ means that x and y *occur* in the guard. The order is not specified, and the arguments of the underlying atom can be any terms over x and y.

Lemma 3.2.1. If $\mathfrak{A} : \Theta$, $t(x)$ is a term over a variable x, $u(y)$ is a term over a variable y, $A(t, u)$ is a binary atom over terms t and u, and $\mathfrak{A}, \mu \vDash A(t, u)$ then $\delta_{\mathfrak{A}}(\mu(x), \mu(y)) \leqslant 1$.

Proof. Let $a = [\![t]\!]_{\mathfrak{A},\mu}$ and $b = [\![u]\!]_{\mathfrak{A},\mu}$. By assumption $\delta_{\mathfrak{A}}(a, b) \leqslant 1$. By definition of $\delta_{\mathfrak{A}}$, $\delta_{\mathfrak{A}}(a, \mu(x)) = 0$ and $\delta_{\mathfrak{A}}(b, \mu(y)) = 0$. So $\delta_{\mathfrak{A}}(\mu(x), \mu(y) \leqslant 1$. □

3.2.3. FO[↓] on transitions

A simple syntactic change can shift all the constraints expressed by a formula towards either the precondition or the postcondition of its models. Consider $\phi(x) := \mathsf{Active}(x) \vee \mathsf{Active}^\star(x)$. $\phi(x)$ requires that x is active at least at some point during the transition. However the starred version of ϕ, $\phi^\star(x) := \mathsf{Active}^\star(x) \vee \mathsf{Active}^\star(x)$, requires x to be active in the postcondition;

Definition (Starred and unstarred formulas)**.** If ϕ is a formula, $\phi^\star$ is ϕ with every symbol from **Dyn** replaced with its counterpart from **Dyn***. Conversely, ϕ° is ϕ with every symbol from **Dyn*** replaced with its counterpart from Dyn*. ■

Formulas may mention symbols in **Stat**, **Dyn** and **Dyn***. A formula which does not mention symbols in **Dyn** is not talking about the precondition of its models. A formula which avoids **Dyn*** ignores the postcondition of its models.

If $\phi = \phi^\star$, then ϕ is a *post* formula. If $\phi = \phi^\circ$, it is a *pre* formula.

Convention. If ϕ is a pre, FO formula and $\mathfrak{S} : \Theta^\circ$, we freely talk about the semantics in ϕ in $\mathfrak{S}$ by taking ϕ to be a formula in the language of Θ°. ■

3.2.4. Runs on formulas

Definition (Runs on formulas)**.** Continuing an earlier definition (p.30), if ϕ is a formula, the runs of ϕ are the runs of $[\![\exists\phi]\!]$. We abuse notation and write $\rho(\phi)$ for $\rho([\![\exists\phi]\!])$. Similarly, if $\mathfrak{S} : \Theta^\circ$ we write $\rho_{\mathfrak{S}}(\phi)$ for $\rho_{\mathfrak{S}}([\![\exists\phi]\!])$. ■

3.2.5. FO characterisation of support and strong support

Recall that a supported transition $\mathfrak{A}$ has its domain covered by $[\![\mathsf{Has}]\!]_{\mathfrak{A}} \cup [\![\mathsf{Has}^\star]\!]_{\mathfrak{A}}$. Supported transitions can be characterized by a universal sentence:

Definition (*Support*)**.** The formula *Support* $:= \forall x.\ \mathsf{Has}(x) \vee \mathsf{Has}^\star(x)$ characterizes supported transitions (no proof needed). ■

Moreover, strongly supported transitions are supported and the interpretations of Has and $\mathsf{Has}^\star$ are 0-closed, which can also be characterized by a universal sentence:

Definition (*Support*$_0$)**.** Let

$$\mathsf{Has}_0 := \forall x.\ \mathsf{Has}(x) \leftrightarrow \bigwedge_{h \in \mathbf{Stat}_{\mathrm{fun}}} \mathsf{Has}(h(x))$$

$$\mathit{Support}_0 := \mathit{Support} \wedge \mathsf{Has}_0 \wedge \mathsf{Has}^\star_0$$

The formula *Support*$_0$ characterizes strongly supported structures (no proof needed). If ϕ is any formula:

$$\phi_{S0} := \phi \wedge \mathit{Support}_0$$

■

3.3. Context-freeness & Retractability

A note on runs and executions We start with a note on the difference between ρ and `exec` to motivate the existence of each notion.

Suppose we start with a definition of *what happens* given a set of rewrite rules `rules` and a starting state $\mathfrak{M}$. This *execution semantics* $\mathtt{exec}_{\mathfrak{M}}(\mathtt{rules})$ is based on an operational pushout construction.

Now suppose there is some algorithm which, given a formula ϕ and a set of patterns P_ϕ induced by ϕ, yields the rules $\mathtt{rules}_{P_\phi}$. To evaluate the correctness of the rules, a reasonable criterion could be

The rewrite steps allowed by $\mathtt{rules}_{P_\phi}$ *are the models of* ϕ.

But that would restrict ϕ too much. To widen the set of acceptable formulas, we choose to weaken the criterion to

Modulo a "nice" starting state, the rewrite steps allowed by $\mathtt{rules}_{P_\phi}$ *are the models of* ϕ.

This naturally leads to defining, in addition to the transition semantics $[\![\phi]\!]$, the *run semantics* $\rho_{\mathfrak{M}}(\phi)$ that describe exactly the traces with starting state $\mathfrak{M}$ composed of models of ϕ.

So `exec` is an idealised execution method working on an input `rules`. To define the correctness of `exec` when given some rules induced by a formula ϕ, we defined a notion of execution *purely* *in terms of the formula*. Now we have a framework to evaluate transformations from formulas to sets of rewrite rules.

Starting state aside, to connect $\rho(\phi)$ and $\texttt{exec}(\texttt{rules}_{P_\phi})$, we will go through the patterns P_ϕ. In the past chapter, we have shown when $\texttt{exec}(\texttt{rules}_{P_\phi}) = \rho(\text{Insts}(P_\phi))$. Now, we will show when $\rho(\text{Insts}(P_\phi)) = \rho(\phi)$.

Context-freeness and retractability Equipped with a language for specifying transitions, we move in this section to the relationship between pattern matches and satisfaction of a formula. We define the dual notions of context-freeness and retractability. The first transports satisfiability forward along a match arrow, the other backward. We then introduce a set of deduction rules, $\vdash_*$, such that $\vdash_*$-provable formulas are both context-free and retractable. Those rules act as a "typing mechanism" for the formulas.

In chapter 2 we described how to relate executions on rules derived from a set of pattern P to the runs on *instances* of those patterns. In the previous section, we described runs on *models of formulas*. This section is a first step towards connecting runs on formulas and runs derived from pattern instances.

We need a brief definition, then we continue with the explanations:

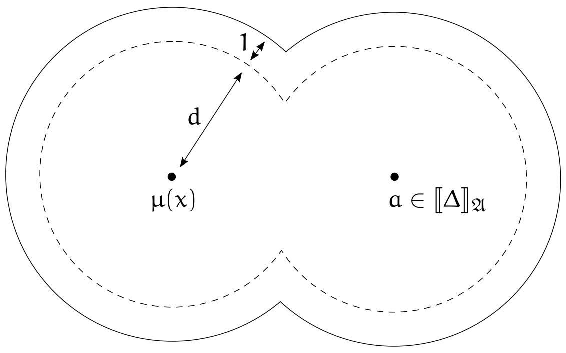


Figure 3.2.: A canonical pattern. $\mathfrak{A}$ is in full outline. $\mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d)$ is in dashed outline. The ball encompasses a radius of d around both the image of μ and elements in $[\![\Delta]\!]_{\mathfrak{A}}$.

Canonical patterns To construct canonical patterns given a formula ϕ and $d \geqslant 0$, we take all models $\mathfrak{A}$ of ϕ that do not go beyond a radius $d+1$ around:

- The interpretation of the variables that made $\mathfrak{A}$ satisfy ϕ.
- The changing elements of $\mathfrak{A}$, that is the elements $a \in [\![\Delta]\!]_{\mathfrak{A}}$.

For each of those, the application condition is the ball of radius d around those same elements. So as illustrated in figure 3.2, the application condition leaves a ring of radius 1 around the edges of the model.

Notation. If $\mathfrak{A} : \Theta$, μ is a function into $\text{dom}(\mathfrak{A})$ and $d \geqslant 0$,

$$\mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d) \coloneqq \mathcal{B}_{\mathfrak{A}}([\![\Delta]\!]_{\mathfrak{A}} \cup \text{Im}(\mu), d)$$

■

Definition (Canonical patterns $\mathcal{P}(\phi, d)$)**.** If ϕ is a formula and $d \geqslant 0$, the canonical patterns $\mathcal{P}(\phi, d)$ are the patterns of the form $(\mathfrak{A}, \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d))$ such that $\mathrm{dom}(\mathfrak{A}) = \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d+1)$ and $\mathfrak{A}, \mu \vDash \phi$. ■

Relationship betwen instances of canonical patterns and models Our goal is to find out when the runs associated with a formula are equal to the executions associated with some set of rules. Under the right conditions, we will show that for a formula ϕ and a starting state $\mathfrak{M}_0$, the right set of rules is

$$\mathtt{rules}(\mathcal{P}(\phi, \|\phi\|))$$

and obtain the following equality:

$$\rho_{\mathfrak{M}_0}(\phi) = \mathtt{exec}_{\mathfrak{M}_0}(\mathtt{rules}(\mathcal{P}(\phi, \|\phi\|))$$

From the previous chapter, we know conditions where, for P a set of patterns,

$$\rho_{\mathfrak{M}_0}(\mathrm{Insts}(P)) = \mathtt{exec}_{\mathfrak{M}_0}(\mathtt{rules}(P))$$

Let $P = \mathcal{P}(\phi, \|\phi\|)$. All we need to complete the equality is

$$\rho_{\mathfrak{M}}(\phi) = \rho_{\mathfrak{M}}(\mathrm{Insts}(\mathcal{P}(\phi, \|\phi\|)))$$

On the left-hand side, the runs are directly built from the models of the formula. On the right-hand side, we start by only looking at *some* models of the formula, then turn these models into patterns, then generate *all* possible instances of those patterns and build runs from those instances.

We will need the following two properties:

1. For the inclusion $\rho_{\mathfrak{M}}(\phi) \subseteq \rho_{\mathfrak{M}}(\mathrm{Insts}(\mathcal{P}(\phi, \|\phi\|)))$, we want any model of ϕ to be an instance of some pattern in $\mathcal{P}(\phi, \|\phi\|)$. The solution will be, given a transition $\mathfrak{A}$, to construct a pattern p and *deduce* $p \in \mathcal{P}(\phi, \|\phi\|)$ from the fact that $\mathfrak{A}$ satisfies ϕ.

2. For the inclusion $\rho_{\mathfrak{M}}(\phi) \supseteq \rho_{\mathfrak{M}}(\mathrm{Insts}(\mathcal{P}(\phi, \|\phi\|)))$, we want *enough* instances of $\mathcal{P}(\phi, \|\phi\|)$ to be models of ϕ. The solution will be, given a pattern p with an instance $\mathfrak{A}$ and some side-conditions, to *deduce* that $\mathfrak{A}$ satisfies ϕ from the fact that $p \in \mathcal{P}(\phi, \|\phi\|)$.

Recall the following general fact: purely existential formulas are preserved by embeddings, and purely universal ones are preserved by retraction along embeddings. See figure 3.3 for an illustration in our setting. We will define two parameterised sets: context-free and retractable formulas. If $(\mathfrak{A}, K)$ is a pattern and $m : (\mathfrak{A}, K) \to \mathfrak{B}$, $\mathfrak{A}$ may satisfy ϕ but $\mathfrak{B}$ may not (and vice-versa). Both directions are important: if a transition $\mathfrak{B}$ is part of a run in $\rho_{\mathfrak{M}}(\phi)$, it will be in a run of $\rho_{\mathfrak{M}}(\mathrm{Insts}(\mathcal{P}(\phi, \|\phi\|)))$ only if some pattern from $\mathcal{P}(\phi, \|\phi\|)$ can instantiate to it. If ϕ is *retractable*, it will be satisfied by a pattern of the right shape. In the other direction, a transition $\mathfrak{A}$ part of a run induced by pattern instances will be in $\rho_{\mathfrak{M}}(\phi)$ only if it satisfies ϕ. If ϕ is *context-free*, it will be satisfied by $\mathfrak{A}$.

In the next section, we will study the dual notions of context-freeness and retractability, defined on formulas, which correspond to maintaining satisfiability by going forwards or backwards along a match, modulo a proviso on the *precondition* of the match target. We will introduce a first system of rules, $\vDash_*$, which acts as a "typing" for formulas: $\vDash_*$-provable formulas are both context-free and retractable.

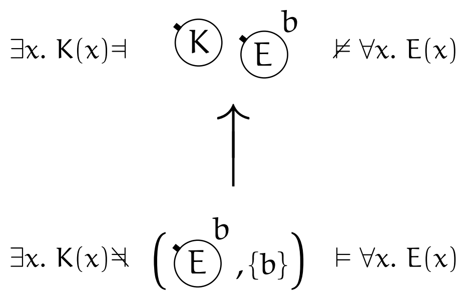


Figure 3.3.: A match and two formulas

After that, there will still be some work left to do: the proviso which guarantees preservation of satisfaction needs to be true for all the elements of a run. We will introduce a restriction of the rules of $\vdash_*$, which we call $\vdash_\rho$. For $\vdash_\rho$-provable formulas, the truth or falsity of the proviso is invariant along runs. So we will show how, modulo some side conditions, a $\vdash_\rho$-provable formula ϕ induces the same runs as the instances of its canonical patterns.

3.3.1. Context-free, retractable formulas

Both notions of context-free and retractable use the following definition. The idea is that the target of a function f may need to verify some proviso *outside* of the image of f, on its *precondition*, but the image itself may be dispensed from verifying the proviso.

Definition (Validity in image complement)**.** For any structure $\mathfrak{B}$, function f that maps to $\mathrm{dom}(\mathfrak{B})$ and any formula I, we write $\mathfrak{B} \vDash \forall(I \vee f)$ to mean that for all $\nu : \langle I \rangle \to \mathrm{dom}(\mathfrak{B})$, $\mathrm{Im}(\nu) \subseteq \mathrm{Im}(f)$ or $\mathfrak{B}, \nu \vDash I$. ■

Given a radius d, a proviso I, two transitions $\mathfrak{A}, \mathfrak{B}$ and a particular application condition K. Suppose $m : (\mathfrak{A}, K) \to B$, and suppose $\mathfrak{B}$ satisfies I at least outside of the image of m. Then ϕ being (I, d)-context-free means that m transports ϕ-satisfaction forwards, and ϕ being (I, d)-retractable means that m transports it backwards.

Definition (Context-freeness)**.** If ϕ is a formula, I is a pre formula, and $d \geqslant 0$, ϕ is (I, d)-*context-free* if for all $\mathfrak{A}, \mathfrak{B}$ and $m : (\mathfrak{A}, \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d)) \to \mathfrak{B}$, if $\mathfrak{A}, \mu \vDash \phi$ and $\mathfrak{B} \vDash \forall(I \vee m)$ then $\mathfrak{B}, m \circ \mu \vDash \phi$. ■

Any suitable d is a *context radius* of (ϕ, I).

Definition (Retractability)**.** If ϕ is a formula, I is a pre formula I, $d \geqslant 0$, ϕ is (I, d)-*retractable* if for all $\mathfrak{A}, \mathfrak{B}, \mu : \langle \phi \rangle \to \mathrm{dom}(\mathfrak{A})$ and $m : (\mathfrak{A}, \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d)) \to \mathfrak{B}$, if $\mathfrak{B}, m \circ \mu \vDash \phi$ and $\mathfrak{B} \vDash \forall(I \vee m)$ then $\mathfrak{A}, \mu \vDash \phi$. ■

Any suitable d is a *retraction radius* of (ϕ, I).

We informally rephrase context-free and retractable in plain english. For some parameters (I, d) and μ, consider any "safe enough" match m from some $\mathfrak{A}$ to some $\mathfrak{B}$, namely:

a) m transports a d-radius ball containing $\mathrm{Im}(\mu)$ and the changes in $\mathfrak{A}$ to $\mathfrak{B}$

b) any tuple from $\mathfrak{B}$ fully outside of $\mathrm{Im}(m)$ satisfies the precondition I.

Then:

- ϕ is context-free if $\mathfrak{A}, \mu \vDash \phi$ implies $\mathfrak{B}, m \circ \mu \vDash \phi$
- ϕ is retractable if $\mathfrak{B}, m \circ \mu \vDash \phi$ implies $\mathfrak{A}, \mu \vDash \phi$

Example. Consider the formulas from figure 3.3. $\forall x.\ \mathsf{E}(x)$ is not context-free for any (I, d), but it is $(\top, 0)$-retractable. Conversely, $\exists x.\ \mathsf{K}(x)$ is not retractable for any (I, d) but it is $(\top, 0)$-context-free. ■

Good clauses

We will soon introduce rules that can build context-free and retractable formulas. Before that we need to define a class of clauses: good clauses. A refinement of good clauses, called great clauses, will come into play later.

Definition (Subliterals of a clause)**.** In a clause $\mathcal{C} = \bigvee_{1 \leqslant i \leqslant n} L_i$, let $\mathcal{L}_{\mathbf{Dyn}}(\mathcal{C})$ be the set of literals of $\mathcal{C}$ that use a symbol from $\mathbf{Dyn}$, and $\mathcal{L}^{\star}_{\mathbf{Dyn}}(\mathcal{C})$ be the set of literals of $\mathcal{C}$ that use a symbol from $\mathbf{Dyn}^{\star}$. ■

Definition (Graph of a clause)**.** Let $G_{\mathcal{C}}$ be the undirected graph with vertices $\langle \mathcal{C} \rangle$, and an edge $\{x, y\}$ iff there is $\neg\mathsf{Link}(t, u)$ or $\neg\mathsf{Link}^{\star}(t, u)$ in $\mathcal{C}$ such that $\{\langle t \rangle, \langle u \rangle\} = \{x, y\}$. ■

Definition (Good clause)**.** A clause $\mathcal{C}$ is *good* if $\mathcal{L}^{\star}_{\mathbf{Dyn}}(\mathcal{C})$ nonempty implies $G_{\mathcal{C}}$ connected. ■

Example. The clause $\mathsf{A}(x) \vee \mathsf{Link}(x, y)$ is good. The clause $\mathsf{A}^{\star}(x) \vee \mathsf{Link}(x, y)$ is not good. The clause $\mathsf{A}^{\star}(x) \vee \neg\mathsf{Link}(x, y)$ is good. ■

Rules $\models_*$ and first theorem

The rest of this subsection displays the rules $\models_*$ and states the main theorem of this section. Then, we list each pair of lemmas that proves the correctness of each rules and conclude with the main proof of the theorem.

In a sequent $\Gamma \models_* \phi$, Γ is a set of clauses $\mathcal{C}_1, \ldots, \mathcal{C}_k$ and ϕ is a formula. The rules $\models_*$ are an intermediate step towards the rules that actually guarantee correct execution. The asterisk $*$ is a little training wheel that will come off eventually.

$$\frac{}{\models_* L}\ \textsc{Literal}_* \qquad\qquad \mathcal{C}\text{ good}\ \frac{}{\mathcal{C}^{\circ} \models_* \forall\mathcal{C}}\ \textsc{Clause}_*$$

$$\frac{\Gamma_1 \models_* \phi_1 \qquad \Gamma_2 \models_* \phi_2}{\Gamma_1, \Gamma_2 \models_* \phi_1 \vee \phi_2}\ \textsc{Disj}_* \qquad\qquad \frac{\Gamma_1 \models_* \phi_1 \qquad \Gamma_2 \models_* \phi_2}{\Gamma_1, \Gamma_2 \models_* \phi_1 \wedge \phi_2}\ \textsc{Conj}_*$$

$$\frac{\Gamma \models_* \phi}{\Gamma \models_* \forall x.\ \alpha(x, y) \to \phi}\ \forall\textsc{Guard}_* \qquad\qquad \frac{\Gamma \models_* \phi}{\Gamma \models_* \exists x.\ \alpha(x, y) \wedge \phi}\ \exists\textsc{Guard}_*$$

$$\frac{\Gamma \models_* \phi}{\Gamma \models_* \downarrow(\phi \wedge \mathit{Support})}\ \textsc{Circum}_*$$

Notation (Clause set to formula)**.** If $\Gamma = \mathcal{C}_1, \ldots, \mathcal{C}_k$ is a set of clauses, $\bigwedge\Gamma = \bigwedge_{1 \leqslant i \leqslant k} \mathcal{C}_i$. ■

The theorem to prove is:

Theorem 3.3.22. If $\Gamma \models_* \phi$ then ϕ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\phi\|)$-context-free and retractable.

The proof will be found in section 3.3.3.

3.3.2. Building context-free and retractable formulas

Weakening

The weakening says that what works with a certain "protection radius" and certain proviso works with a larger radius and an even stronger proviso.

The proofs up to paragraph **Context-free clauses** are routine and can be found in the appendix, section A.1. Skip to page 43 for the more interesting part.

Lemma 3.3.1. If ϕ is (I, d)-context-free then ϕ is (I', d')-context-free for any $d' \geqslant d$, $I' \vDash I$.

Lemma 3.3.2. If ϕ is (I, d)-retractable then ϕ is (I', d')-retractable for any $d' \geqslant d$, $I' \vDash I$.

There is no need for an explicit weakening rule in $\vdash_*$ since $\textsc{disj}_*$ and $\textsc{conj}_*$ are multiplicative.

Quantifier-free formulas

Lemma 3.3.3. A quantifier-free formula ϕ is $(\top, 0)$-context-free.

Lemma 3.3.4. A quantifier-free formula ϕ is $(\top, 0)$-retractable.

Boolean composition

Lemma 3.3.5. If ϕ_1 is (I, d_1)-context-free (resp. retractable) and ϕ_2 is (I, d_2)-context-free (resp. retractable) then $\phi_1 \wedge \phi_2$ is $(I, \max(d_1, d_2))$-context-free (resp. retractable).

Lemma 3.3.6. If ϕ_1 is (I, d_1)-context-free (resp. retractable) and ϕ_2 is (I, d_2)-context-free (resp. retractable) then $\phi_1 \vee \phi_2$ is $(I, \max(d_1, d_2))$-context-free (resp. retractable).

Universal quantification

Lemma 3.3.7. If ϕ is (I, d)-context-free and $\alpha(x, y)$ is a guard, then $\forall x.\ \alpha(x, y) \rightarrow \phi$ is $(I, d + 1)$-context-free.

Lemma 3.3.8. If ϕ is (I, d)-retractable then for any x, $\forall x.\ \phi$ is (I, d)-retractable.

Existential quantification

Lemma 3.3.9. If ϕ is (I, d)-context-free then for any x, $\exists x.\ \phi$ is (I, d)-retractable.

Lemma 3.3.10. If ϕ is (I, d)-retractable and $\alpha(x, y)$ is a guard then $\exists x.\ \alpha(x, y) \wedge \phi$ is $(I, d + 1)$-retractable.

Context-free clauses

The main difficulty is the CLAUSE$_*$ rule for the context-free part. To transport a universal constraint *forward* along a morphism, we do need to some conditions to hold on the destination.

The following lemma simply rephrases the definition of $[\![\Delta]\!]$ as the *changes* of a transition by asserting that things outside of $[\![\Delta]\!]$ do not change indeed.

Lemma 3.3.11. If $L(t, u)$ is a pre literal ($t = u$ if L unary) and $\mathfrak{A}, \mu$ is such that $\delta_{\mathfrak{A}}([\![\langle t\rangle]\!]_{\mathfrak{A},\mu}, [\![\Delta]\!]_{\mathfrak{A}}) > 0$ then $\mathfrak{A}, \mu \vDash L(t, u) \leftrightarrow L^\star(t, u)$.

Proof. Let S be the symbol used in $L(t, u)$. If $S \notin$ **Dyn**, $L(t, u) = L^\star(t, u)$ so we are done. Otherwise, since $\delta_{\mathfrak{A}}([\![\langle t\rangle]\!]_{\mathfrak{A},\mu}, [\![\Delta]\!]_{\mathfrak{A}}) > 0$, we have $\delta_{\mathfrak{A}}([\![t]\!]_{\mathfrak{A},\mu}, [\![\Delta]\!]_{\mathfrak{A}}) > 0$. Since $\mathfrak{A}, \mu \vDash \Delta L(t, u)$ implies $[\![t]\!]_{\mathfrak{A},\mu} \in [\![\Delta]\!]_{\mathfrak{A}}$, we are done. $\square$

The following lemma gives a key property of good clauses. Consider $G_{\mathcal{C}}$. Assuming it is connected, $\mathcal{C}$ can only be false when the interpretation of $\langle\mathcal{C}\rangle$ fits in a small enough ball.

Lemma 3.3.12. If $\mathcal{C}$ is a clause such that $G_{\mathcal{C}}$ is connected and $\mathfrak{A}, \mu$ is such that $\mathfrak{A}, \mu \nvDash L$ for every literal $L \in \mathcal{C}$ of the form $\neg\mathsf{Link}(t, u)$ or $\neg\mathsf{Link}^\star(t, u)$ then $d_{\mathfrak{A}}(a, b) < |\mu(\langle\mathcal{C}\rangle)|$ for every $a, b \in \mu(\langle\mathcal{C}\rangle)$.

Proof. Let G be the undirected graph with vertices $\mu(\langle\mathcal{C}\rangle)$ and an edge $\{a, b\}$ whenever $d_{\mathfrak{A}}(a, b) \leqslant 1$. By assumption, $G_{\mathcal{C}}$ is connected. We show that G is connected for any two $a, b \in G$ by induction on $\delta_{G_{\mathcal{C}}}(\mu^{-1}(a), \mu^{-1}(b))$.

If the distance is 0, then $a = b$ and we are done. Otherwise, there is y such that $\delta_{G_{\mathcal{C}}}(\mu^{-1}(a), y) = k$, $\delta_{G_{\mathcal{C}}}(y, \mu^{-1}(b)) = 1$ and by induction hypothesis, there is a path in G from a to $\mu(y)$. Now by definition of $G_{\mathcal{C}}$, $\mathfrak{A}, \mu \vDash \mathsf{Link}(t, u) \vee \mathsf{Link}^\star(t, u)$ with $\{\langle t\rangle, \langle u\rangle\} = \{y, m^{-1}(b)\}$. So $d_{\mathfrak{A}}(\mu(y), b) \leqslant 1$. So there is a path from a to b in G. So G is connected, so its diameter is $\leqslant |\mu(\langle\mathcal{C}\rangle)|$. $\square$

This lemma gives a simple consequence of the definition of a match: in a match, changes in the target have a preimage in the changes of the source.

Lemma 3.3.13. If $m : (\mathfrak{A}, K) \to \mathfrak{B}$, then $[\![\Delta]\!]_{\mathfrak{B}} \subseteq m([\![\Delta]\!]_{\mathfrak{A}})$.

Proof. For every $A \in$ **Dyn**, every $\mathbf{b} \in [\![\Delta A]\!]_{\mathfrak{B}}$ and every $b \in \mathbf{b}$, $b \in [\![\Delta]\!]_{\mathfrak{B}}$. So $b \in m(\mathfrak{A})$ by definition of a match. So $m^{-1}(b)$ is well-defined. So by definition of an embedding, $m^{-1}(b) \in [\![\Delta A]\!]_{\mathfrak{A}}$. $\square$

The following lemma explains the importance of good clauses and the presence of a proviso in the definition of context-free. By default, universal formulas are very much not context-free – the new context might betray the universal specification. Here we see that for a good clause, it is enough to check that the match target satisfies the *pre* version of the clause. Note that the context around the image of a match contains no changes, so it should be enough to know that the *pre* version of the clause is satisfied. The $|\langle\mathcal{C}\rangle|$ radius makes sure that the new context doesn't come too close to the changes (which are all defined by the domain of the match).

Lemma 3.3.14. If $\mathcal{C}$ is good and $\forall\mathbf{x}.\ \mathcal{C}$ is a sentence, $\forall\mathbf{x}.\ \mathcal{C}$ is $(\mathcal{C}^\circ, |\langle\mathcal{C}\rangle| - 1)$-context-free.

Proof. Let $d := |\langle\mathcal{C}\rangle|$. Take any $\mathfrak{A}, \mathfrak{B}, m : (\mathfrak{A}, \mathcal{B}^{\Delta,\emptyset}_{\mathfrak{A}}(d-1)) \to \mathfrak{B}$ such that $\mathfrak{A} \vDash \forall\mathbf{x}.\ \mathcal{C}$, and for all $\nu : \langle\mathcal{C}^\circ\rangle \to \mathrm{dom}(\mathfrak{B})$, $\mathrm{Im}(\nu) \subseteq m(\mathfrak{A})$ or $\mathfrak{B}, \nu \vDash \mathcal{C}^\circ$. We want $\mathfrak{B} \vDash \forall\mathbf{x}.\ \mathcal{C}$.

Let $\eta : \langle\mathcal{C}\rangle \to \mathrm{dom}(\mathfrak{B})$. If $\mathrm{Im}(\eta) \subseteq m(\mathfrak{A})$, let $\mu := m^{-1} \circ \eta$. By assumption, $\mathfrak{A}, \mu \vDash \mathcal{C}$. Since $\mathcal{C}$ is quantifier-free and m is an embedding, $\mathfrak{B}, \eta \vDash \mathcal{C}$ and we are done.

Otherwise there is $a \in \mathrm{Im}(\eta) \setminus m(\mathfrak{A})$. Moreover $\langle\mathcal{C}\rangle = \langle\mathcal{C}^\circ\rangle$ (trivially), so $\eta : \langle\mathcal{C}^\circ\rangle \to \mathrm{dom}(\mathfrak{B})$. So by assumption on $\mathfrak{A}, \mathfrak{B}, m$ and η, $\mathfrak{B}, \eta \vDash \mathcal{C}^\circ$.

If $\mathcal{L}^{\star}_{\mathsf{Dyn}}(\mathfrak{C})$ is empty, then $\mathfrak{C} = \mathfrak{C}^{\circ}$ and we are done.

Otherwise, by definition of good, $G_{\mathfrak{C}}$ is connected. We get the following inclusions by the lemmas indicated below:

$$\mathcal{B}_{\mathfrak{B}}([\![\Delta]\!]_{\mathfrak{B}}, d) \underset{\text{lemma 3.3.13}}{\subseteq} \mathcal{B}_{\mathfrak{B}}(m([\![\Delta]\!]_{\mathfrak{A}}), d) \underset{\text{lemma 2.4.5}}{\subseteq} m(\mathfrak{A})$$

Since $a \notin m(\mathfrak{A})$, $\delta_{\mathfrak{B}}(a, [\![\Delta]\!]_{\mathfrak{B}}) > d$.

If there is a literal L in $\mathfrak{C}$ of the form $\neg\mathsf{Link}^{\star}(t, u)$ or $\neg\mathsf{Link}(t, u)$ such that $\mathfrak{B}, \eta \vDash L$, then $\mathfrak{B}, \eta \vDash \mathfrak{C}$ and we are done.

If there is no such L then by lemma 3.3.12 $d_{\mathfrak{B}}(b, b') < |\mathrm{Im}(\eta)| \leqslant |\langle\mathfrak{C}\rangle| = d$ for every $b, b' \in \mathrm{Im}(\eta)$. Since $\delta_{\mathfrak{B}}(a, [\![\Delta]\!]_{\mathfrak{B}}) > d$, we get $\delta_{\mathfrak{B}}(b, [\![\Delta]\!]_{\mathfrak{B}}) > 0$ for every $b \in \mathrm{Im}(\eta)$. Consider any L in $\mathfrak{C}$ such that $\mathfrak{B}, \eta \vDash L^{\circ}$ (by assumption, there is at least one).

If L is not a post literal, then $L^{\circ} = L$ so $\mathfrak{B}, \eta \vDash \mathfrak{C}$.

Otherwise, since $G_{\mathfrak{C}}$ is connected, we apply lemma 3.3.11 and get $\mathfrak{B}, \eta \vDash L$, so $\mathfrak{B}, \eta \vDash \mathfrak{C}$.

□

Example. In figure 3.4, we illustrate good clauses with an example of a match between two structures $\mathfrak{A}$ and $\mathfrak{B}$. $\mathfrak{A}$ satisfies $F_1 := \forall x.\ \mathsf{On}(x) \vee \mathsf{Active}^{\star}(x)$. We explain why $\mathfrak{B}$ does as well.

By CLAUSE$_*$, $\mathsf{On}(x) \vee \mathsf{Active}(x) \vDash_{*} F_1$. So by lemma 3.3.14, F_1 is $(\mathsf{On} \vee \mathsf{Active}, 0)$-context-free. Note that $\mathfrak{A}, x \mapsto a \nvDash \mathsf{On}(x) \vee \mathsf{Active}(x)$, yet $\mathfrak{A} \vDash F_1$. The elements of $\mathrm{dom}(\mathfrak{B})$ that are not in $\mathrm{dom}(\mathfrak{A})$ *do* satisfy $\mathsf{On}(x) \vee \mathsf{Active}(x)$. Crucially, b is not involved in a change, and thus is still Active in the postcondition. Thus, we get $\mathfrak{B} \vDash F_1$. Generally speaking, in the rule CLAUSE$_*$, having $\mathfrak{C}$ be good ensures that, if a tuple satisfies $\mathfrak{C}^{\circ}$ and is not involved in a change, then it satisfies $\mathfrak{C}$. ■

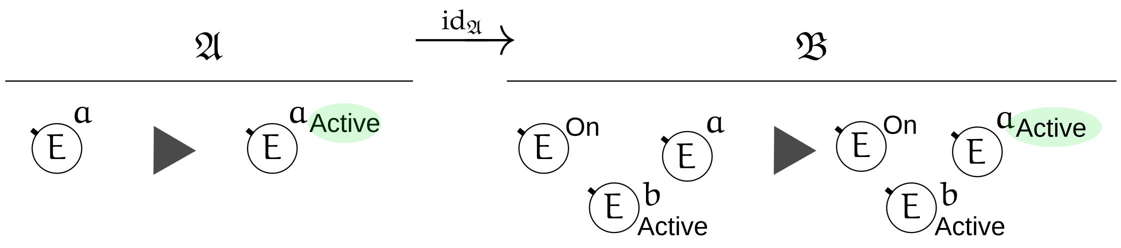


Figure 3.4.: $\mathfrak{A}$ and $\mathfrak{B}$ satisfy $F_1 := \forall x.\ \mathsf{On}(x) \vee \mathsf{Active}^{\star}(x)$. Changes highlighted in green.

Example. In figure 3.5, we see a more complex example. The transition $\mathfrak{A}$ vacuously satisfies the formula $F_2 := \forall xy.\ \mathsf{Link}(x, y) \to \mathsf{E}(x) \to \mathsf{On}^{\star}(x)$ (with $\mathsf{E} \in$ **Stat**). The match $\mathrm{id}_{\mathfrak{A}} : (\mathfrak{A}, \mathcal{B}^{\Delta,\emptyset}_{\mathfrak{A}}(1)) \to \mathfrak{B}$ ensures $\mathfrak{B} \vDash F_2$ provided $\mathfrak{B}, \mu \vDash \mathsf{Link}(x, y) \to \mathsf{E}(x) \to \mathsf{On}(x)$ whenever $\mathrm{Im}(\mu) \nsubseteq \mathrm{Im}(\mathrm{id}_{\mathfrak{A}})$. a gains an edge (through the match, *not* during the transition) and is allowed to since it is not in $[\![\Delta]\!]_{\mathfrak{A}}$. The proviso only requires $a \in [\![\mathsf{On}]\!]_{\mathfrak{A}}$: since a is not involved in a change, that implies $a \in [\![\mathsf{On}^{\star}]\!]_{\mathfrak{A}}$. However b is involved in a change, and thus cannot become linked through the match. ■

Minimisation

We are in FO[↓], not just FO, so we must study how to build context-free and retractable minimised formulas. Here is the main idea, informally: suppose ϕ is both context-free and retractable. Is $\downarrow\phi$ context-free?

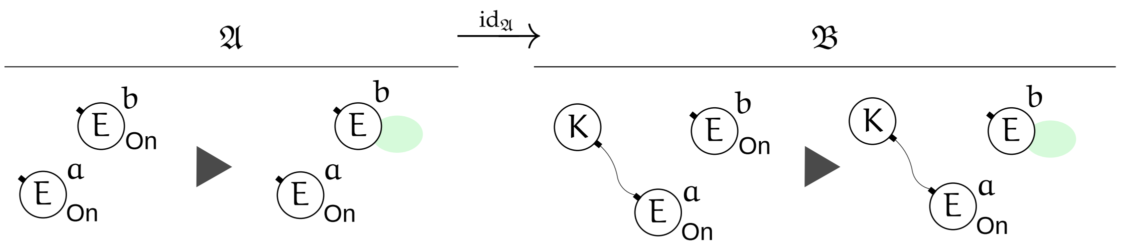


Figure 3.5.: $\mathfrak{A}$ and $\mathfrak{B}$ satisfy $\mathsf{F}_2 := \forall xy.\ \mathsf{Link}(x,y) \to \mathsf{E}(x) \to \mathsf{On}^\star(x)$

Take some $m : (\mathfrak{A}, K) \to \mathfrak{B}$. We know $\mathfrak{A} \vDash_{\downarrow} \phi$, so (modulo a proviso and some facts about K) $\mathfrak{B} \vDash \phi$. The hard question is: is $\mathfrak{B}$ minimal among models ϕ?

Suppose it is not, that is: suppose there is $\mathfrak{B}' \lhd \mathfrak{B}$ such that $\mathfrak{B}' \vDash \phi$. Lemma 3.3.18 will show that under the right circumstances, we can build a transition $\mathfrak{A}' \lhd \mathfrak{A}$ such that there is a "nice" match from $\mathfrak{A}'$ to $\mathfrak{B}'$. Now, since we supposed that ϕ was not only context-free but also retractable, $\mathfrak{B}' \vDash \phi$ yields $\mathfrak{A}' \vDash \phi$, which contradicts the minimality of $\mathfrak{A}$ among models of ϕ. That reasoning made precise can be found in lemma 3.3.19.

There is also a technical requirement that we only consider supported models, due to the definition of $\trianglelefteq$. An alternative design (but with fairly wide-ranging consequences) would be to remove the $\mathrm{dom}(\mathfrak{A}) \subseteq \mathrm{dom}(\mathfrak{B})$ in the definition of $\mathfrak{A} \trianglelefteq \mathfrak{B}$.

The same reasoning works if one asks whether $\downarrow\phi$ is retractable, as shown in lemmas 3.3.20 and 3.3.21.

The following lemma simply rests on the fact that the change order cannot touch preconditions, so pre formulas are not affected by it.

Lemma 3.3.15. If ψ is a pre, quantifier-free formula, $\mathfrak{A} \trianglelefteq \mathfrak{B}$ and $\nu : \langle\psi\rangle \to \mathrm{dom}(\mathfrak{A})$, then $\mathfrak{A}, \nu \vDash \psi$ iff $\mathfrak{B}, \nu \vDash \psi$.

Proof. Trivial induction on the structure of ψ, base cases for atoms use the fact that, by definition of $\trianglelefteq$, $[\![\mathbf{Dyn}]\!]_{\mathfrak{A}} = [\![\mathbf{Dyn}]\!]_{\mathfrak{B}}$ and $[\![\mathbf{Stat}]\!]_{\mathfrak{A}} = [\![\mathbf{Stat}]\!]_{\mathfrak{B}} \upharpoonright \mathrm{dom}(\mathfrak{A})$. □

This lemma gives an immediate consequence of the fact that $\mathsf{Has}(x) \vee \mathsf{Has}^\star(x)$ is good. This is an ad hoc proof of something $\vDash_*$ is designed to prove, but we need it before we can prove all of $\vDash_*$ correct.

Lemma 3.3.16. If ϕ is (I, d)-context-free (resp. (I, d)-retractable), then $\phi \wedge \mathit{Support}$ is $(I \wedge \mathsf{Has}, d)$-context-free (resp. $(I \wedge \mathsf{Has}, d)$-retractable).

Proof. For retractable, lemmas 3.3.4, 3.3.8 and 3.3.2 immediately give the conclusion. For context-free, we note that $\mathsf{Has}(x) \vee \mathsf{Has}^\star(x)$ is good and $|\langle \mathsf{Has}(x) \vee \mathsf{Has}^\star(x)\rangle| = 1$. For context-free, by lemma 3.3.14, *Support* is $(\mathsf{Has}, 0)$-context-free. By lemmas 3.3.5 and 3.3.1, we are done. □

This simple lemma gives another way to state the effect of application conditions on 0-embeddings.

Lemma 3.3.17. If $m : (\mathfrak{A}, K) \to \mathfrak{B}$, $A \in \mathbf{Dyn}$, $\mathbf{b} \in [\![A]\!]_{\mathfrak{B}}$, and $b \in m(K)$ for some $b \in \mathbf{b}$ then there is $\mathbf{a} \in [\![A]\!]_{\mathfrak{A}}$ such that $m(\mathbf{a}) = \mathbf{b}$.

Proof. If $b \in m(K)$ then by definition of a match and $d_{\mathfrak{B}}(b, b') \leqslant 1$ for every $b' \in \mathbf{b}$, $m^{-1}(\mathbf{b})$ is well-defined. So by definition of an embedding, $m^{-1}(\mathbf{b}) \in [\![A]\!]_{\mathfrak{A}}$. □

We now move to the main lemmas for proving that minimisation does not impact context-freeness and retractability. Illustrations are included.

$$\begin{tikzcd}
 & & \mathfrak{B} & \\
 & & & \\
\mathfrak{A} \arrow[uurr, "m"] & & \triangleleft & \mathfrak{B}' \arrow[uul, phantom, "\triangleleft"] \\
\triangleleft & & & \\
 & \exists\, \mathfrak{A}' \arrow[uurr, dashed, "m \restriction m^{-1}(\mathfrak{B}')"'] & &
\end{tikzcd}$$

Figure 3.6.: Illustration of the statement of lemma 3.3.18

Context-freeness of $\downarrow\phi$ This next pair of lemmas shows how to obtain context-freeness of $\downarrow\phi$ from both retractability and context-freeness of ϕ. Figure 3.6 illustrates the first one.

Lemma 3.3.18. If $\mathfrak{B} : \Theta$ is supported, $\mathfrak{B}' \lhd \mathfrak{B}$, $\mathfrak{A} : \Theta$, $\mu : X \to \mathrm{dom}(\mathfrak{A})$ for X some set of variables, and $m : (\mathfrak{A}, \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d)) \to \mathfrak{B}$ such that $m(\mu(X)) \subseteq \mathrm{dom}(\mathfrak{B}')$, then there are $\mathfrak{A}'$ and $m' : (\mathfrak{A}', \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}'}(d)) \to \mathfrak{B}'$ such that $\mathfrak{A}' \lhd \mathfrak{A}$ and $m' = m \restriction m^{-1}(\mathfrak{B}')$.

Proof. We start by defining $\mathfrak{A}'$.

$$\begin{aligned}
\mathrm{dom}(\mathfrak{A}') &:= \mathrm{dom}(\mathfrak{A}) \setminus m^{-1}(\mathrm{dom}(\mathfrak{B}) \setminus \mathrm{dom}(\mathfrak{B}')) && \\
[\![A]\!]_{\mathfrak{A}'} &:= [\![A]\!]_{\mathfrak{A}} && \text{for all } A \in \mathbf{Dyn} \\
[\![T]\!]_{\mathfrak{A}'} &:= [\![T]\!]_{\mathfrak{A}} \restriction \mathrm{dom}(\mathfrak{A}') && \text{for all } T \in \mathbf{Stat} \\
[\![\Delta A]\!]_{\mathfrak{A}'} &:= [\![\Delta A]\!]_{\mathfrak{A}} \setminus m^{-1}([\![\Delta A]\!]_{\mathfrak{B}} \setminus [\![\Delta A]\!]_{\mathfrak{B}'}) && \text{for all } A \in \mathbf{Dyn}
\end{aligned}$$

We show that $\mathfrak{A}'$ is well-defined, i.e. that the above definitions induce a single structure and that the domain covers the elements mentioned in the relations.

- The definitions for every $A \in \mathbf{Dyn}$ of $[\![A]\!]_{\mathfrak{A}'}$ and $[\![\Delta A]\!]_{\mathfrak{A}'}$ induce

$$[\![A^\star]\!]_{\mathfrak{A}'} = ([\![\Delta A]\!]_{\mathfrak{A}'} \setminus [\![A]\!]_{\mathfrak{A}'}) \cup ([\![A]\!]_{\mathfrak{A}'} \setminus [\![\Delta A]\!]_{\mathfrak{A}'})$$

- Suppose there is $\mathbf{a} \in [\![A]\!]_{\mathfrak{A}'}$. $\mathbf{a} \in [\![A]\!]_{\mathfrak{A}}$ by definition of $\unlhd$. $m(\mathbf{a}) \in [\![A]\!]_{\mathfrak{B}}$ by definition of an embedding. $m(\mathbf{a}) \in [\![A]\!]_{\mathfrak{B}'}$ by definition of $[\![A]\!]_{\mathfrak{B}'}$. So for every $a \in \mathbf{a}$, $m(a) \in \mathrm{dom}(\mathfrak{B}')$, and thus by definiton of $\mathrm{dom}(\mathfrak{A}')$, $a \in \mathrm{dom}(\mathfrak{A}')$.
- Suppose there is $\mathbf{a} \in [\![A^\star]\!]_{\mathfrak{A}'}$. Wlog $\mathbf{a} \notin [\![A]\!]_{\mathfrak{A}'}$ (see previous case), so $\mathbf{a} \in [\![\Delta A]\!]_{\mathfrak{A}'}$. $\mathbf{a} \in [\![\Delta A]\!]_{\mathfrak{A}}$ by definition of $\unlhd$. $m(\mathbf{a}) \in [\![\Delta A]\!]_{\mathfrak{B}}$ by definition of an embedding. So, by definition of $[\![\Delta A]\!]_{\mathfrak{A}'}$, $m(\mathbf{a}) \in [\![\Delta A]\!]_{\mathfrak{B}'}$. So $a \in \mathrm{dom}(\mathfrak{B}')$ for every $a \in \mathbf{a}$, and thus by definition of $\mathrm{dom}(\mathfrak{A}')$, $a \in \mathrm{dom}(\mathfrak{A}')$.
- The case $\mathbf{a} \in [\![T]\!]_{\mathfrak{B}'}$ is trivial.

Now we show $\mathfrak{A}' \lhd \mathfrak{A}$. By definition $\mathfrak{A}' \unlhd \mathfrak{A}$. Moreover $\mathfrak{B}' \lhd \mathfrak{B}$, so $[\![\Delta A]\!]_{\mathfrak{B}'} \subset [\![\Delta A]\!]_{\mathfrak{B}}$ for some $A \in \mathbf{Dyn}$ or $\mathrm{dom}(\mathfrak{B}') \subset \mathrm{dom}(\mathfrak{B})$.

- If there is $b \in \mathrm{dom}(\mathfrak{B}) \setminus \mathrm{dom}(\mathfrak{B}')$, by definition of $\lhd$ we have $b \notin [\![\mathsf{Has}]\!]_{\mathfrak{B}}$. Since $\mathfrak{B}$ is supported, $b \in [\![\Delta\mathsf{Has}]\!]_{\mathfrak{B}}$, and so by lemma 3.3.13 there is $a \in [\![\Delta\mathsf{Has}]\!]_{\mathfrak{A}}$ such that $m(a) = b$. So by definition of $\mathrm{dom}(\mathfrak{A}')$, $a \notin \mathrm{dom}(\mathfrak{A}')$ and thus $\mathfrak{A}' \lhd \mathfrak{A}$.

- If there is $\mathbf{b} \in [\![\Delta A]\!]_{\mathfrak{B}} \setminus [\![\Delta A]\!]_{\mathfrak{B}'}$ for some $A \in$ **Dyn**, by lemma 3.3.13 there is $\mathbf{a} \in [\![\Delta A]\!]_{\mathfrak{A}}$ such that $m(\mathbf{a}) = \mathbf{b}$. By definition of $[\![\Delta A]\!]_{\mathfrak{A}'}$, $\mathbf{a} \notin [\![\Delta A]\!]_{\mathfrak{A}'}$ and thus $\mathfrak{A}' \lhd \mathfrak{A}$.

With $m' = m \restriction m^{-1}(\mathfrak{B}')$, we need $m' : (\mathfrak{A}', \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d)) \to \mathfrak{B}'$, i.e. that $m^{-1}(\mathfrak{B}') = \mathrm{dom}(\mathfrak{A}')$, $\mu(X) \subseteq \mathrm{dom}(\mathfrak{A}')$, that m is an embedding, that $\mathcal{B}_{\mathfrak{B}'}(m'(\mathcal{B}_{\mathfrak{A}'}^{\Delta,\mu}(d)), 1) \subseteq m'(\mathfrak{A}')$ and that $[\![\Delta]\!]_{\mathfrak{B}'} \subseteq m'(\mathfrak{A}')$.

- We show $m^{-1}(\mathfrak{B}') = \mathrm{dom}(\mathfrak{A}')$. If $a \in m^{-1}(\mathfrak{B}')$ then $m(a) \in m(\mathfrak{A})$, so by definition of $\mathrm{dom}(\mathfrak{A}')$, $a \in \mathrm{dom}(\mathfrak{A}')$. In the other direction, if $a \in \mathrm{dom}(\mathfrak{A}')$ then $a \in \mathrm{dom}(\mathfrak{A})$ as well by definition of $\unlhd$, so $a \in m^{-1}(\mathfrak{B})$. Thus by definition of $\mathrm{dom}(\mathfrak{A}')$, $a \in m^{-1}(\mathfrak{B}')$.
- $\mu(X) \subseteq \mathrm{dom}(\mathfrak{A}')$ follows from $m^{-1}(\mathfrak{B}') \subseteq \mathrm{dom}(\mathfrak{A}')$ and the assumption $m(\mu(X)) \subseteq \mathrm{dom}(\mathfrak{B}')$.
- Since m is an embedding and by definition of $\mathfrak{A}'$, $m' : \mathfrak{A}' \to \mathfrak{B}'$ is an embedding. To show that it is a 0-embedding, it suffices to show that for all $a \in \mathrm{dom}(\mathfrak{B}')$, function symbol $h \in$ **Stat**, $b \in \mathrm{Im}(m')$ such that $(a, b) \in [\![h]\!]_{\mathfrak{B}'}$, we have $a \in \mathrm{Im}(m)$. By definition of $\mathfrak{B}'$, that premise implies $\delta_{\mathfrak{B}}(a, b) = 0$, so, since m is a 0-embedding and m' is a restriction of m, we get $a \in \mathrm{Im}(m)$.
- Let $b \in \mathrm{dom}(\mathfrak{B}')$ be at distance 1 (in $\mathfrak{B}'$) from $m'(a)$ with $a \in \mathcal{B}_{\mathfrak{A}'}^{\Delta,\mu}(d)$. Since $[\![\Delta]\!]_{\mathfrak{A}'} \subseteq [\![\Delta]\!]_{\mathfrak{A}}$ and by definition of m, $b \in m(\mathfrak{A})$. Suppose $b \notin \mathrm{dom}(\mathfrak{A}')$: by definition of $\mathrm{dom}(\mathfrak{A}')$, $b \notin \mathrm{dom}(\mathfrak{B}')$. Absurd.
- Finally by $[\![\Delta]\!]_{\mathfrak{B}'} \subseteq [\![\Delta]\!]_{\mathfrak{B}}$ and lemma 3.3.13, $[\![\Delta]\!]_{\mathfrak{B}'} \subseteq m([\![\Delta]\!]_{\mathfrak{A}})$. It is easy to see that $[\![\Delta]\!]_{\mathfrak{A}'} = [\![\Delta]\!]_{\mathfrak{A}} \setminus m^{-1}([\![\Delta]\!]_{\mathfrak{B}} \setminus [\![\Delta]\!]_{\mathfrak{B}'})$, so $[\![\Delta]\!]_{\mathfrak{B}'} \subseteq m([\![\Delta]\!]_{\mathfrak{A}'})$. Since $\mathrm{dom}(\mathfrak{A}') \subseteq m^{-1}(\mathfrak{B}')$, $[\![\Delta]\!]_{\mathfrak{B}'} \subseteq m'([\![\Delta]\!]_{\mathfrak{A}'})$.

$\square$

Recall the reasoning at the beginning of the **minimisation** paragraph above and keep in mind figure 3.6, which summarises the previous lemma. We use the contrapositive of the previous lemma to prove the next one.

Lemma 3.3.19. If ϕ is (I, d_c)-context-free and (I, d_r)-retractable, then $\downarrow(\phi \wedge \mathit{Support})$ is $(I \wedge \mathsf{Has}, \max(d_c, d_r))$-context-free.

Proof. Let d_c be any context radius and d_r be any retraction radius for ϕ, I, and let $d := \max(d_c, d_r)$. Consider $\mathfrak{A}, \mathfrak{B}, \mu : \langle\phi\rangle \to \mathrm{dom}(\mathfrak{A})$ and $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d)) \to \mathfrak{B}$ such that $\mathfrak{A}, \mu \vDash \downarrow (\phi \wedge \mathit{Support})$ and $\mathfrak{B} \vDash \forall((I \wedge \mathsf{Has}) \vee m)$. We want $\mathfrak{B}, m \circ \mu \vDash \downarrow (\phi \wedge \mathit{Support})$. Assume not. By lemma 3.3.16, $\phi \wedge \mathit{Support}$ is $(I \wedge \mathsf{Has}, d_c)$-context-free. Since $d \geqslant d_c$, by lemma 3.3.1 $\mathfrak{B}, m \circ \mu \vDash \phi \wedge \mathit{Support}$. So there is $\mathfrak{B}' \lhd \mathfrak{B}$ such that $\mathfrak{B}', m \circ \mu \vDash \phi \wedge \mathit{Support}$.

Since $\mathfrak{B}$ is supported and $m \circ \mu$ maps to $\mathrm{dom}(\mathfrak{B}')$, by lemma 3.3.18, there is $\mathfrak{A}' \lhd \mathfrak{A}$ and $m' : (\mathfrak{A}', \mathcal{B}_{\mathfrak{A}'}^{\Delta,\mu}(d)) \to \mathfrak{B}'$ such that $m \restriction m^{-1}(\mathfrak{B}') = m'$. We show $\mathfrak{A}', \mu \vDash \phi \wedge \mathit{Support}$, which contradicts $\mathfrak{A}, \mu \vDash \downarrow (\phi \wedge \mathit{Support})$:

Take any $\nu : \langle I \wedge \mathsf{Has}\rangle \to \mathrm{dom}(\mathfrak{B}')$. Since $I \wedge \mathsf{Has}$ is a pre, quantifier free formula and $\mathfrak{B}' \unlhd \mathfrak{B}$, by lemma 3.3.15 $\mathfrak{B}, \nu \vDash I \wedge \mathsf{Has}$ implies $\mathfrak{B}', \nu \vDash I \wedge \mathsf{Has}$. Otherwise, by assumption $\mathrm{Im}(\nu) \subseteq \mathrm{Im}(m)$. Since $\mathrm{Im}(\nu) \subseteq \mathrm{dom}(\mathfrak{B}')$, $\mathrm{Im}(\nu) \subseteq \mathrm{Im}(m \restriction m^{-1}(\mathfrak{B}')) = m'(\mathfrak{A}')$. So $\mathfrak{B}' \vDash \forall((I \wedge \mathsf{Has}) \vee m')$. By lemma 3.3.16, $\phi \wedge \mathit{Support}$ is $(I \wedge \mathsf{Has}, d_r)$-retractable. Since $d \geqslant d_r$, by lemma 3.3.2 we get $\mathfrak{A}', \mu \vDash \phi \wedge \mathit{Support}$.

$\square$

$$\begin{tikzcd}
 & \mathfrak{B} & \\
\mathfrak{A} \arrow[ur, "\mathfrak{m}"] & & \exists\ \mathfrak{B}' \arrow[ul, phantom, "\lhd"] \\
 & \mathfrak{A}' \arrow[ul, phantom, "\lhd"] \arrow[ur, dashed, "\mathfrak{m} \upharpoonright \mathfrak{m}^{-1}(\mathfrak{B}')"'] &
\end{tikzcd}$$

Figure 3.7.: Illustration of the statement of lemma 3.3.20

Retractability of $\downarrow\phi$ This next pair of lemmas shows how to obtain retractability of $\downarrow\phi$ from both retractability and context-freeness of ϕ. Figure 3.7 illustrates the first one.

Lemma 3.3.20. If $\mathfrak{A} : \Theta$ is supported, $\mathfrak{A}' \lhd \mathfrak{A}$, $\mathfrak{B} : \Theta$, $\mu : X \to \mathrm{dom}(\mathfrak{A}')$ for X some set of variables, and $\mathfrak{m} : (\mathfrak{A}, \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d)) \to \mathfrak{B}$, then there are $\mathfrak{B}'$ and $\mathfrak{m}' : (\mathfrak{A}', \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}'}(d)) \to \mathfrak{B}'$ such that $\mathfrak{B}' \lhd \mathfrak{B}$ and $\mathfrak{m}' = \mathfrak{m} \upharpoonright \mathfrak{m}^{-1}(\mathfrak{B}')$.

Proof. We start by defining $\mathfrak{B}'$.

$$\begin{aligned}
\mathrm{dom}(\mathfrak{B}') &:= \mathrm{dom}(\mathfrak{B}) \setminus \mathfrak{m}(\mathrm{dom}(\mathfrak{A}) \setminus \mathrm{dom}(\mathfrak{A}')) & \\
\llbracket A \rrbracket_{\mathfrak{B}'} &:= \llbracket A \rrbracket_{\mathfrak{B}} & \text{for all } A \in \mathbf{Dyn} \\
\llbracket T \rrbracket_{\mathfrak{B}'} &:= \llbracket T \rrbracket_{\mathfrak{B}} \upharpoonright \mathrm{dom}(\mathfrak{B}') & \text{for all } T \in \mathbf{Stat} \\
\llbracket \Delta A \rrbracket_{\mathfrak{B}'} &:= \llbracket \Delta A \rrbracket_{\mathfrak{B}} \setminus \mathfrak{m}(\llbracket \Delta A \rrbracket_{\mathfrak{A}} \setminus \llbracket \Delta A \rrbracket_{\mathfrak{A}'}) & \text{for all } A \in \mathbf{Dyn}
\end{aligned}$$

We show that $\mathfrak{B}'$ is well-defined, i.e. that the above definitions induce a single structure and that the domain covers the elements mentioned in the relations.

- The definitions for every $A \in \mathbf{Dyn}$ of $\llbracket A \rrbracket_{\mathfrak{B}'}$ and $\llbracket \Delta A \rrbracket_{\mathfrak{B}'}$ induce
$$\llbracket A^\star \rrbracket_{\mathfrak{B}'} = (\llbracket \Delta A \rrbracket_{\mathfrak{B}'} \setminus \llbracket A \rrbracket_{\mathfrak{B}'}) \cup (\llbracket A \rrbracket_{\mathfrak{B}'} \setminus \llbracket \Delta A \rrbracket_{\mathfrak{B}'})$$
- Suppose there is $b \in \mathfrak{m}(\mathfrak{A})$ and $\mathbf{b}$ such that $b \in \mathbf{b}$ and $\mathbf{b} \in \llbracket A \rrbracket_{\mathfrak{B}'}$. If $\mathfrak{m}(\mathbf{a}) = \mathbf{b}$ for some $\mathbf{a} \in \llbracket A \rrbracket_{\mathfrak{A}}$, then $\mathfrak{m}^{-1}(b) \in \mathrm{dom}(\mathfrak{A}')$ by definition of $\unlhd$. Otherwise by lemma 3.3.17 (taking the contrapositive) $\mathfrak{m}^{-1}(b) \notin \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d)$. So $\mathfrak{m}^{-1}(b) \notin \llbracket \Delta \rrbracket_{\mathfrak{A}}$. Since $\mathfrak{A}$ is supported, $\mathfrak{m}^{-1}(b) \in \llbracket \mathsf{Has} \rrbracket_{\mathfrak{A}}$, and so $\mathfrak{m}^{-1}(b) \in \mathrm{dom}(\mathfrak{A}')$ by definition of $\unlhd$.
- Suppose that for some $b \in \mathfrak{m}(\mathfrak{A})$, $b \in \mathbf{b} \in \llbracket A^\star \rrbracket_{\mathfrak{B}'}$. Wlog $\mathbf{b} \notin \llbracket A \rrbracket_{\mathfrak{B}'}$ (see previous case). So $\mathbf{b} \in \llbracket \Delta A \rrbracket_{\mathfrak{B}'}$. By lemma 3.3.13, $\mathbf{b} \in \mathfrak{m}(\llbracket \Delta A \rrbracket_{\mathfrak{A}})$. So by definition of $\llbracket \Delta A \rrbracket_{\mathfrak{B}'}$, $\mathbf{b} \in \mathfrak{m}(\llbracket \Delta A \rrbracket_{\mathfrak{A}'})$. So $b \in \mathfrak{m}(\mathfrak{A}')$.
- The case $\mathbf{b} \in \llbracket T \rrbracket_{\mathfrak{B}'}$ is trivial.

Now we show $\mathfrak{B}' \lhd \mathfrak{B}$. By definition $\mathfrak{B}' \unlhd \mathfrak{B}$. Moreover $\mathfrak{A}' \lhd \mathfrak{A}$, so $\llbracket \Delta A \rrbracket_{\mathfrak{A}'} \subset \llbracket \Delta A \rrbracket_{\mathfrak{A}}$ for some $A \in \mathbf{Dyn}$ or $\mathrm{dom}(\mathfrak{A}') \subset \mathrm{dom}(\mathfrak{A})$. In any case $\mathfrak{B}' \lhd \mathfrak{B}$.

Finally we show $\mathfrak{m} \upharpoonright \mathfrak{m}^{-1}(\mathfrak{B}') : (\mathfrak{A}', \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d)) \to \mathfrak{B}'$.

With $\mathfrak{m}' = \mathfrak{m} \upharpoonright \mathfrak{m}^{-1}(\mathfrak{B}')$, we need $\mathfrak{m}' : (\mathfrak{A}', \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d)) \to \mathfrak{B}'$, i.e. that $\mathfrak{m}^{-1}(\mathfrak{B}') = \mathrm{dom}(\mathfrak{A}')$, that $\mu(X) \subseteq \mathrm{dom}(\mathfrak{A}')$, that $\mathfrak{m}$ is an embedding, that $\mathcal{B}_{\mathfrak{B}'}(\mathfrak{m}'(\mathcal{B}^{\Delta,\mu}_{\mathfrak{A}'}(d)), 1) \subseteq \mathfrak{m}'(\mathfrak{A}')$ and that $\llbracket \Delta \rrbracket_{\mathfrak{B}'} \subseteq \mathfrak{m}'(\mathfrak{A}')$.

- We show $m^{-1}(\mathfrak{B}') = \mathrm{dom}(\mathfrak{A}')$. If $a \in m^{-1}(\mathfrak{B}')$ then $m(a) \in m(\mathfrak{A})$, and by definition of $\mathrm{dom}(\mathfrak{B}')$, $m(a) \notin m(\mathrm{dom}(\mathfrak{A}) \setminus \mathrm{dom}(\mathfrak{A}'))$ so $a \in \mathrm{dom}(\mathfrak{A}')$. In the other direction, if $a \in \mathrm{dom}(\mathfrak{A}')$ then $a \in \mathrm{dom}(\mathfrak{A})$ as well by definition of $\trianglelefteq$, so $m(a) \in \mathrm{dom}(\mathfrak{B}')$ by definition of $\mathrm{dom}(\mathfrak{B}')$.

- $\mu(X) \subseteq \mathrm{dom}(\mathfrak{A}')$ by assumption on $\mathfrak{A}$

- Since m is an embedding and by definition of $\mathfrak{B}'$, $m' : \mathfrak{A}' \to \mathfrak{B}'$ is an embedding. To show that it is a 0-embedding, it suffices to show that for all $a \in \mathrm{dom}(\mathfrak{B}')$, function symbol $h \in$ **Stat**, $b \in \mathrm{Im}(m')$ such that $(a, b) \in [\![h]\!]_{\mathfrak{B}'}$, we have $a \in \mathrm{Im}(m)$. By definition of $\mathfrak{B}'$, that premise implies $\delta_{\mathfrak{B}}(a, b) = 0$, so, since m is a 0-embedding and m' is a restriction of m, we get $a \in \mathrm{Im}(m)$.

- Let $b \in \mathrm{dom}(\mathfrak{B}')$ be at distance 1 (in $\mathfrak{B}'$) from $m'(a)$ with $a \in \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}'}(d)$. Since $[\![\Delta]\!]_{\mathfrak{A}'} \subseteq [\![\Delta]\!]_{\mathfrak{A}}$ and by definition of m, $b \in m(\mathfrak{A})$. Suppose $b \notin \mathrm{dom}(\mathfrak{A}')$: by definition of $\mathrm{dom}(\mathfrak{B}')$, $b \notin \mathrm{dom}(\mathfrak{B}')$. Absurd.

- Finally by $[\![\Delta]\!]_{\mathfrak{B}'} \subseteq [\![\Delta]\!]_{\mathfrak{B}}$ and lemma 3.3.13, $[\![\Delta]\!]_{\mathfrak{B}'} \subseteq m([\![\Delta]\!]_{\mathfrak{A}})$. It is easy to see that $[\![\Delta]\!]_{\mathfrak{B}'} = [\![\Delta]\!]_{\mathfrak{B}} \setminus m([\![\Delta]\!]_{\mathfrak{A}} \setminus [\![\Delta]\!]_{\mathfrak{A}'})$, so $[\![\Delta]\!]_{\mathfrak{B}'} \subseteq m([\![\Delta]\!]_{\mathfrak{A}'})$. Since $\mathrm{dom}(\mathfrak{A}') \subseteq m^{-1}(\mathfrak{B}')$, $[\![\Delta]\!]_{\mathfrak{B}'} \subseteq m'([\![\Delta]\!]_{\mathfrak{A}'})$.

□

Recall the reasoning at the beginning of the **minimisation** paragraph above and keep in mind figure 3.7, which summarises the previous lemma. We use the contrapositive of the previous lemma to prove the next one.

Lemma 3.3.21. If ϕ is (I, d_c)-context-free and (I, d_r)-retractable, then $\downarrow(\phi \wedge \mathit{Support})$ is $(I \wedge \mathsf{Has}, \max(d_c, d_r))$-retractable.

Proof. Let $d := \max(d_c, d_r)$. Consider $\mathfrak{A}$, $\mathfrak{B}, \mu : \langle\phi\rangle \to \mathrm{dom}(\mathfrak{A})$ and $m : (\mathfrak{A}, \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d)) \to \mathfrak{B}$ such that $\mathfrak{B}, m \circ \mu \vDash\downarrow(\phi \wedge \mathit{Support})$ and $\mathfrak{B} \vDash \forall((I \wedge \mathsf{Has}) \vee m)$. We want $\mathfrak{A}, \mu \vDash\downarrow(\phi \wedge \mathit{Support})$. Assume not. By lemma 3.3.16, $\phi \wedge \mathit{Support}$ is $(I \wedge \mathsf{Has}, d_r)$-retractable. Since $d \geqslant d_r$, by lemma 3.3.2 $\mathfrak{A}, \mu \vDash \phi \wedge \mathit{Support}$. So there is $\mathfrak{A}' \triangleleft \mathfrak{A}$ such that $\mathfrak{A}', \mu \vDash \phi \wedge \mathit{Support}$. Note that this implies $\mathrm{Im}(\mu) \subseteq \mathrm{dom}(\mathfrak{A}')$.

Since $\mathfrak{A}$ is supported, by lemma 3.3.20, there is $\mathfrak{B}' \triangleleft \mathfrak{B}$ and $m' : (\mathfrak{A}', \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}'}(d)) \to \mathfrak{B}'$ such that $m \upharpoonright m^{-1}(\mathfrak{B}') = m'$. We show $\mathfrak{B}', m \circ \mu \vDash \phi \wedge \mathit{Support}$, which contradicts $\mathfrak{B}, m \circ \mu \vDash\downarrow (\phi \wedge \mathit{Support})$:

Take any $\nu : \langle I \wedge \mathsf{Has}\rangle \to \mathrm{dom}(\mathfrak{B}')$. Since $I \wedge \mathsf{Has}$ is a pre, quantifier free formula and $\mathfrak{B}' \trianglelefteq \mathfrak{B}$, by lemma 3.3.15 $\mathfrak{B}, \nu \vDash I \wedge \mathsf{Has}$ implies $\mathfrak{B}', \nu \vDash I \wedge \mathsf{Has}$. Otherwise, by assumption $\mathrm{Im}(\nu) \subseteq \mathrm{Im}(m)$. Since $\mathrm{Im}(\nu) \subseteq \mathrm{dom}(\mathfrak{B}')$, $\mathrm{Im}(\nu) \subseteq \mathrm{Im}(m \upharpoonright m^{-1}(\mathfrak{B}')) = m'(\mathfrak{A}')$. So $\mathfrak{B}' \vDash \forall((I \wedge \mathsf{Has}) \vee m')$. By lemma 3.3.16, $\phi \wedge \mathit{Support}$ is $(I \wedge \mathsf{Has}, d_c)$-context-free. Since $d \geqslant d_c$, by lemma 3.3.1 we get $\mathfrak{B}', m' \circ \mu \vDash \phi \wedge \mathit{Support}$.

Since $m \upharpoonright m^{-1}(\mathfrak{B}') = m'$ and $\mathrm{Im}(\mu) \subseteq \mathrm{dom}(\mathfrak{A}')$, $m \circ \mu = m' \circ \mu$, so $\mathfrak{B}', m \circ \mu \vDash \phi$.

□

3.3.3. Correctness of the $\vDash_*$ rules

Theorem 3.3.22 is a direct consequence of the lemmas above.

Theorem 3.3.22. If $\Gamma \vDash_* \phi$ then ϕ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\phi\|)$-context-free and retractable.

Proof. By induction on the derivation.

- LITERAL$_*$: By lemmas 3.3.3 and 3.3.1 L is $(\mathsf{Has}, 0)$-context-free, by lemmas 3.3.4 and 3.3.2 L is $(\mathsf{Has}, 0)$-retractable.

- CLAUSE$_*$: Lemma 3.3.14 together with the proviso on CLAUSE$_*$ implies that $\forall\mathcal{C}$ is $(\mathcal{C}^\circ, |\langle\mathcal{C}\rangle| - 1)$-context-free. Since $\|\forall\mathcal{C}\| \geqslant |\langle\mathcal{C}\rangle| - 1$ (trivial), by lemma 3.3.1 $\forall\mathcal{C}$ is $(\mathcal{C}^\circ \wedge \mathsf{Has}, \|\forall\mathcal{C}\|)$-context-free. $\forall\mathcal{C}$ is $(\mathcal{C}^\circ \wedge \mathsf{Has}, \|\forall\mathcal{C}\|)$-retractable by direct application of lemmas 3.3.4, 3.3.8 and 3.3.2.

- DISJ$_*$: By induction hypothesis each ϕ_i is $(\bigwedge\Gamma_i \wedge \mathsf{Has}, \|\phi_i\|)$-context-free and retractable, and $\|\phi_1 \vee \phi_2\| = \max(\|\phi_1\|, \|\phi_2\|)$ so by lemmas 3.3.1, 3.3.2 and 3.3.6, $\phi_1 \vee \phi_2$ is $(\bigwedge\Gamma_1, \Gamma_2 \wedge \mathsf{Has}, \|\phi_1 \vee \phi_2\|)$-context-free and retractable.

- CONJ$_*$: By induction hypothesis each ϕ_i is $(\bigwedge\Gamma_i \wedge \mathsf{Has}, \|\phi_i\|)$-context-free and retractable, and $\|\phi_1 \wedge \phi_2\| = \max(\|\phi_1\|, \|\phi_2\|)$ so by lemmas 3.3.1, 3.3.2 and 3.3.5, $\phi_1 \vee \phi_2$ is $(\bigwedge\Gamma_1, \Gamma_2 \wedge \mathsf{Has}, \|\phi_1 \wedge \phi_2\|)$-context-free and retractable.

- $\forall$GUARD$_*$: Let $\psi := \forall x.\ \alpha(x, y) \to \phi$. By induction hypothesis ϕ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\phi\|)$-context-free and retractable.

  For context-free: By lemma 3.3.7, and since $\|\psi\| = \|\phi\| + 1$, ψ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\psi\|)$-context-free.

  For retractable: Since $\neg\alpha(x, y)$ is quantifier-free, by lemmas 3.3.4, 3.3.2 and 3.3.6, $\alpha(x, y) \to \phi$ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\psi\|)$-retractable, so by lemma 3.3.8 so is ψ.

- $\exists$GUARD$_*$: Let $\psi := \exists x.\ \alpha(x, y) \wedge \phi$. By induction hypothesis ϕ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\phi\|)$-context-free and retractable.

  For retractable: By lemma 3.3.10, and since $\|\psi\| = \|\phi\| + 1$, ψ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\psi\|)$-retractable.

  For context-free: Since $\alpha(x, y)$ is quantifier-free, by lemmas 3.3.3, 3.3.1 and 3.3.5, $\alpha(x, y) \wedge \phi$ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\psi\|)$-context-free, so by lemma 3.3.9 so is ψ.

- CIRCUM$_*$: Let $\psi :=\downarrow(\phi \wedge \mathit{Support})$. By induction hypothesis, ϕ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\phi\|)$-context-free and retractable. By lemma 3.3.21, ψ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\phi\|)$-retractable (lemma 3.3.21 was applied to $\bigwedge\Gamma \wedge \mathsf{Has}$ and we then rewrote $\bigwedge\Gamma \wedge \mathsf{Has} \wedge \mathsf{Has}$ to $\bigwedge\Gamma \wedge \mathsf{Has}$); since $\|\psi\| \geqslant \|\phi\|$, by lemma 3.3.2 ψ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\psi\|)$-retractable. Similarly, by lemmas 3.3.19 and 3.3.1, ψ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\psi\|)$-context-free.

□

We have defined a class of formulas which survive going through matches modulo some side conditions. This is a good first step, but we need more: we need the side conditions themselves to survive throughout entire runs. That will be the focus of the next section.

3.4. Runs of context-free and retractable formulas

In this section, we refine the deduction rules $\vdash_*$ into $\vdash_\rho$, which lets us prove the main theorem of this chapter: if ϕ is $\vdash_\rho$-provable and the starting state is "nice", then the runs of the models of ϕ are exactly the executions of the rules of the canonical patterns of ϕ. Recall that, by the explanations in section 3.3 (page 39), we already have a correspondence between the execution on a set of models and the denotational runs on a set of models. So it now suffices to show that the runs of the instances of the canonical patterns of ϕ are exactly the executions of the rules of of the canonical patterns of ϕ.

Context-freeness and retractability are defined modulo some formula I on the match target. But this formula may not stay true at every step of the run. We start by refining good clauses to great clauses, and we show why they help maintain that invariant I. Then we introduce the deduction rules $\vdash_\rho$, explain how $\vdash_\rho$ is a restriction of $\vdash_*$ and motivate the differences. Then we state the main theorem before we go about proving it.

First, we give an overview of how that proof will go. Section 3.4.4 contains two key lemmas.

Runs on instances included in run on models Lemma 3.4.6 will use context-freeness, and asserts that runs on pattern instances are included in runs on models of ϕ. The idea is: if ϕ is context-free modulo some I which only requires checking a small enough area, and ϕ ensures I on the postcondition (meaning ϕ ensures $I^\star$), then from a "nice" starting point $\mathfrak{M}_0$ we:

1. Take the first transition $\mathfrak{A}_1$ in a run on the *instances* of the canonical patterns of ϕ (so a run in $\rho_{\mathfrak{M}_0}(\mathsf{Insts}(P))$).
2. Because $\mathfrak{M}_0$ is "nice", the precondition of $\mathfrak{A}_1$ satisfies I (modulo details).
3. The transition is an instance of a pattern $(\mathfrak{B}_1, K_1)$ such that $\mathfrak{B}$ satisfies ϕ (by definition). So by context-freeness of ϕ, $\mathfrak{A}_1$ satisfies ϕ as well.
4. Now that $\mathfrak{A}_1$ satisfies ϕ, a run on the models of ϕ (so a run in $\rho_{\mathfrak{M}_0}(\phi)$) can begin with $\mathfrak{A}_1$.
5. Also, $\mathfrak{A}_1$ satisfying ϕ means that its postcondition satisfies I (by assumption).
6. So the precondition of $\mathfrak{A}_2$, the next transition in the run, also satisfies I (by definition of a run).
7. Moreover, that next transition is also an instance of a pattern $(\mathfrak{B}_2, K_2)$ such that (...)
8. And so on.

Runs on models included in run on instances Lemma 3.4.7 uses retractability, and asserts that the runs on models of ϕ are included in runs on pattern instances. The idea is: if ϕ is retractable modulo some I which only requires checking a small enough area, and ϕ ensures I on the postcondition, then from a "nice" starting $\mathfrak{M}_0$ point we:

1. Take the first transition $\mathfrak{A}_1$ in a run on the *models* of ϕ (so a run in $\rho_{\mathfrak{M}_0}(\phi)$.
2. Because $\mathfrak{M}_0$ is "nice", the precondition of that transition satisfies I (modulo details).
3. We define a special pattern p_1 which has a match into the $\mathfrak{A}_1$. Since ϕ is retractable, the first projection of p_1 satisfies ϕ as well. Because p_1 has a special shape, this means p_1 is one of the canonical patterns of ϕ.
4. Since $\mathfrak{A}_1$ is an instance of p_1, a run on the instances of canonical patterns of ϕ (so a run in $\rho_{\mathfrak{M}_0}(\mathsf{Insts}(P))$) can begin with $\mathfrak{A}_1$.
5. We knew that $\mathfrak{A}_1$ satisfies ϕ, so its postcondition satisfies I (by assumption).
6. So the precondition of $\mathfrak{A}_2$, the next transition in the run, also satisfies I (by definition of a run).

7. Moreover, another special pattern p_2 with a match into $\mathfrak{A}_2$ can be constructed, and (...)

8. And so on.

3.4.1. Overview of the $\vDash_{\rho}$ rules and main theorem

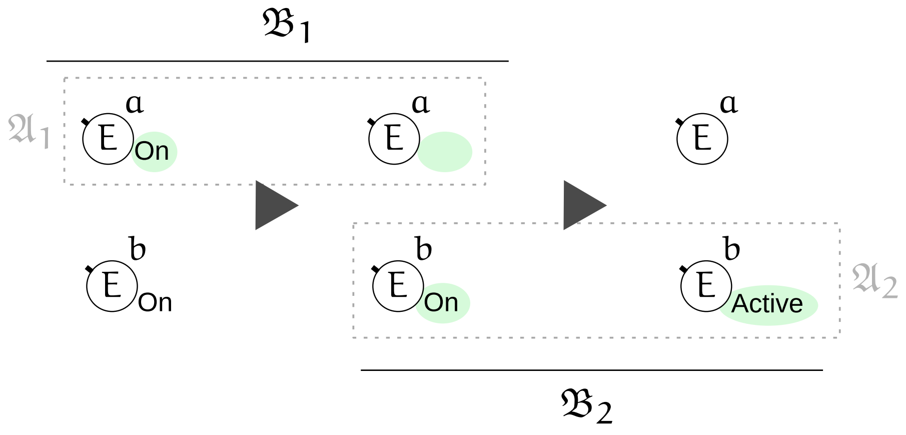


Figure 3.8.: $\mathfrak{B}_1$ satisfies F_1 while $\mathfrak{B}_2$ does not. Changes highlighted in green.

Example. We show a $\vDash_{*}$-provable formula which does *not* remain satisfied throughout a run based on its canonical patterns.

Recall the formula $F_1 := \forall x.\ \mathsf{On}(x) \vee \mathsf{Active}^{\star}(x)$ from figure 3.4. With $\mathcal{C} = \mathsf{On}(x) \vee \mathsf{Active}^{\star}(x)$, we have $\mathcal{C}^{\circ} \vDash_{*} F_1$. So F_1 is $(\mathcal{C}^{\circ}, 0)$-context-free and retractable.

Consider figure 3.8. It represents two transitions, $\mathfrak{B}_1, \mathfrak{B}_2$, where the interpretation in $\mathfrak{B}_1$ of **Dyn*** is equal to the interpretation in $\mathfrak{B}_2$ of **Dyn** – that is, the middle column is the postcondition of $\mathfrak{B}_1$ and the precondition of $\mathfrak{B}_2$. In addition, the identity is a match from $\mathfrak{A}_1$ to $\mathfrak{B}_1$ and from $\mathfrak{A}_2$ to $\mathfrak{B}_2$. Each $\mathfrak{A}_i$ is defined by a grey, dashed outline. Now consider:

- In $\mathfrak{B}_1$, a ceases to be On. Shown in a dotted box is the transition $\mathfrak{A}_1$. There is a match $\mathrm{id}_{\mathfrak{A}_1} : (\mathfrak{A}_1, \mathcal{B}^{\Delta,\emptyset}_{\mathfrak{A}_1}(0)) \to \mathfrak{B}_1$. Since $\mathfrak{B}_1, x \mapsto b \vDash \mathcal{C}^{\circ}$ we obtain $\mathfrak{B}_1 \vDash F_1$ (by context-freeness).

- In $\mathfrak{B}_2$, b goes from being On to being Active. Shown in a dotted box is the transition $\mathfrak{A}_2$. There is a match $\mathrm{id}_{\mathfrak{A}_2} : (\mathfrak{A}_2, \mathcal{B}^{\Delta,\emptyset}_{\mathfrak{A}_2}(0)) \to \mathfrak{B}_2$. But $\mathfrak{B}_1, x \mapsto a \nvDash \mathcal{C}^{\circ}$. And indeed: $\mathfrak{B}_1 \nvDash F_1$.

The issue here is that $\mathfrak{B}_2, x \mapsto a \nvDash \mathcal{C}^{\circ}$. Or, said differently, the issue is that $\mathfrak{B}_1, x \mapsto a \nvDash \mathcal{C}^{\star}$. With a an element of $\mathfrak{B}_1$, $a \notin [\![\mathsf{On}^{\star}]\!]_{\mathfrak{B}_1}$ is fine. But the postcondition of $\mathfrak{B}_1$ becomes the precondition of $\mathfrak{B}_2$, and $a \notin [\![\mathsf{On}]\!]_{\mathfrak{B}_2}$ is *not* fine.

As a result, while the run $\mathfrak{B}_1, \mathfrak{B}_2$ is *not* in $\rho(F_1)$, it is in $\rho(\mathsf{Insts}(\mathcal{P}(F_1, 1)))$.

Informally, our goal is now to find for which formulas we could deduce $\mathfrak{B}_1 \vDash \forall \mathcal{C}^{\star}$ from $\mathfrak{B}_1 \vDash F_1$. That way, the proviso $\mathcal{C}^{\circ}$ will propagate the run $\mathfrak{B}_1, \mathfrak{B}_2$.

■

Great clauses

Great clauses are a restriction on good clauses.

Definition (Great clause). A clause $\mathcal{C}$ is *great* if $\mathcal{L}_{\mathsf{Dyn}}(\mathcal{C})$ nonempty implies that there is some $L \in \mathcal{L}_{\mathsf{Dyn}}(\mathcal{C})$ such that $\overline{L^\star} \in \mathcal{L}^\star_{\mathsf{Dyn}}(\mathcal{C})$. and $G_{\mathcal{C}}$ is connected. ■

Of course, any great clause is also good.

The first condition ($\mathcal{L}_{\mathsf{Dyn}}(\mathcal{C})$ nonempty implies some $L \in \mathcal{L}_{\mathsf{Dyn}}(\mathcal{C})$ such that $\overline{L^\star} \in \mathcal{L}^\star_{\mathsf{Dyn}}(\mathcal{C})$) solves the requirement outlined in the introduction of this section: if $\mathcal{C}$ depends on the precondition at all, then $\mathcal{C}^\circ$ is equivalent to $\top$. So it becomes very easy for a precondition to satisfy $\mathcal{C}^\circ$.

The second condition ($G_{\mathcal{C}}$ connected) is a technical requirement to fulfill the requirements in the definition of context-free and retractable: the $\mathfrak{B} \models \forall(I \vee f)$ part of the definition needs $I(x_1, \ldots, x_k)$ to be true by default when the interpretations for $x_1, \ldots, x_k$ are too far apart.

Example. The clause $A(x) \vee \mathsf{Link}(x, y)$ is good but not great. Same for $A(x) \vee \neg\mathsf{Link}(x, y)$: good but not great. $\neg A(x) \vee B(x) \vee A^\star(x)$ is great. $\neg\mathsf{Link}^\star(x, y)$ is great. ■

Rules $\models_\rho$

The rules $\models_\rho$ are a restriction of $\models_*$. The small ρ replaces the training wheel $*$ and should evoke runs, because $\models_\rho$-provable formulas will be well-behaved on runs.

$$\frac{}{\models_\rho L}\ \textsc{Literal}_\rho \qquad\qquad \mathcal{C}\ \text{great}\ \frac{}{\mathcal{C}^\circ \models_\rho \forall\mathcal{C}}\ \textsc{Clause}_\rho$$

$$\frac{\Gamma \models_\rho \phi_1 \qquad \Gamma \models_\rho \phi_2}{\Gamma \models_\rho \phi_1 \vee \phi_2}\ \textsc{Disj}_\rho \qquad\qquad \frac{\Gamma_1 \models_\rho \phi_1 \qquad \Gamma_2 \models_\rho \phi_2}{\Gamma_1, \Gamma_2 \models_\rho \phi_1 \wedge \phi_2}\ \textsc{Conj}_\rho$$

$$\frac{\models_\rho \phi}{\models_\rho \forall x.\ \alpha(x, y) \to \phi}\ \forall\textsc{Guard}_\rho \qquad\qquad \frac{\models_\rho \phi}{\models_\rho \exists x.\ \alpha(x, y) \wedge \phi}\ \exists\textsc{Guard}_\rho$$

$$\frac{\Gamma \models_\rho \phi}{\Gamma \models_\rho \downarrow(\phi \wedge \mathit{Support})}\ \textsc{Circum}_\rho$$

Here are the differences between $\models_*$ and $\models_\rho$:

- The proviso of $\textsc{Clause}_\rho$ is "$\mathcal{C}$ great" (it is "$\mathcal{C}$ good" in $\textsc{Clause}_*$)
- $\textsc{Disj}_\rho$ is additive ($\textsc{Disj}_*$ is multiplicative)
- The context in the premise of $\forall\textsc{Guard}_\rho$ and $\exists\textsc{Guard}_\rho$ must be $\emptyset$ (it can be any Γ in $\forall\textsc{Guard}_*$ and $\exists\textsc{Guard}_*$)

The move from good to great was explained earlier.

We give some intuition behind the move from a multiplicative $\textsc{Disj}_*$ to an additive $\textsc{Disj}_\rho$: essentially, weakening does not hold in $\models_\rho$ (while it does in $\models_*$). One way to see it is that in $\models_*$ the context Γ describes a sufficient condition for a match to carry satisfaction (forward and backward). So strengthening Γ was free. Now, Γ should be an *invariant* in the sense that it should always carry over to the next transition in a run. So it can't be strengthened arbitrarily – or it becomes that much harder to prove in the next step. A multiplicative version of $\textsc{Clause}_\rho$ would require proving too strong a conclusion (based on Γ_1, Γ_2) from too weak a premise (based on $\phi_1 \vee \phi_2$). In the case of $\textsc{Conj}_\rho$, Γ_1, Γ_2 happens to be exactly be the weakest context that can prove $\phi_1 \wedge \phi_2$.

An empty context in $\forall\textsc{guard}_\rho$ and $\exists\textsc{guard}_\rho$ is required for similar reasons. For instance, in the case of $\forall\textsc{guard}_\rho$, keeping the same context Γ makes it too strong because the formula $\forall x.\ \alpha(x,y) \to \phi$ might be true while ϕ is not (for some interpretation). It may be possible to find richer contexts that carry enough information to expand the conditions where $\forall\textsc{guard}_\rho$ and $\exists\textsc{guard}_\rho$ apply, but we would lose an important feature of the current rules: since $\textsc{disj}_\rho$ is additive, we can prove formulas of the form $\downarrow(F_1 \wedge G) \vee \downarrow(F_2 \wedge G)$ as long as F_1, F_2 are proved without $\textsc{clause}_\rho$. If $\forall\textsc{guard}_\rho$ and $\exists\textsc{guard}_\rho$ added context information depending on $\alpha(x,y)$, they would not be allowed in the proofs of F_1 and F_2. This feature has practical applications in chapter 5.

Of course, $\Gamma \vdash_\rho \phi$ implies $\Gamma \vdash_* \phi$.

Example. The formula $\forall x.\ \mathsf{On}(x)$ is $\vdash_*$-provable but not $\vdash_\rho$-provable. ■

Example. The formula F_1 shown at the beginning of this section is $\vdash_*$-provable (which means it is context-free and retractable for some radii, but not $\vdash_\rho$-provable. As shown on figure 3.8, it should not be provable: the goal of $\vdash_\rho$ is to build formulas that remain satisfied throughout executions; figure 3.8 shows two sucessive pattern matches where each pattern satisfies F_1 but the target of the 2nd match does not satisfy F_1. ■

Overview of main theorem

To state the theorem, we need the following notions:

Definition (Initial states)**.** If I a pre, quantifier-free formula, a structure $\mathfrak{S} : \Theta^\circ$ is in $\mathrm{Init}(I)$ if $\mathfrak{S}$ is the empty structure or $\mathfrak{S} \models \forall I$. ■

The theorem to prove is:

Theorem 3.5.1. If $\Gamma \vdash_\rho \phi$ and $\mathfrak{M} \in \mathrm{Init}(\bigwedge\Gamma \wedge \mathsf{Has})$, then

$$\rho_{\mathfrak{M}}(\phi_{S0}) = \mathtt{exec}_{\mathfrak{M}}(\mathtt{rules}(\mathcal{P}(\phi_{S0}, \|\phi_{S0}\|)))$$

The proof is in section 3.5.

3.4.2. Run inclusion

Lemma 3.4.2 neatly packs up some sufficient conditions for a set of runs to be included in another. We start with an obvious fact:

Lemma 3.4.1. If A, B are sets of transitions such that $A \subseteq B$ then $\rho(A) \subseteq \rho(B)$

Proof. Let $r \in \rho(A)$. For all $\mathfrak{A}_i, \mathfrak{A}_{i+1} \in r$, $\mathrm{Post}(\mathfrak{A}_i) = \mathrm{Pre}(\mathfrak{A}_{i+1})$ by assumption on r and $\mathfrak{A}_i, \mathfrak{A}_{i+1} \in B$ by assumption on A and B. So $r \in \rho(B)$ by definition of $\rho(B)$. □

The idea behind this lemma is to explicitly state that some invariant is needed; here the set S is a set of states, and run inclusion happens with the invariant shifting from one state to the next.

Lemma 3.4.2. If $\mathfrak{S} : \Theta^\circ$, M, N are two sets of transitions, S a set of states, $\mathfrak{S} \in S$ and for all supported $\mathfrak{A} \in M$, $\mathrm{Pre}(\mathfrak{A}) \in S$ implies $\mathfrak{A} \in N$ and $\mathrm{Post}(\mathfrak{A}) \in S$, then $\rho_{\mathfrak{S}}(M) \subseteq \rho_{\mathfrak{S}}(N)$.

Proof. Let $r := \mathfrak{A}_1, \ldots \in \rho_{\mathfrak{S}}(M)$. By induction on i, we show $\mathrm{Pre}(\mathfrak{A}_i) \in S$ and $\mathrm{Post}(\mathfrak{A}_i) \in S$ for all $A_i \in r$.

If $i = 1$ we have $\mathsf{Pre}(\mathfrak{A}_1) = \mathfrak{S}$ by definition of $\rho_{\mathfrak{S}}(M)$ so by assumption on $\mathfrak{S}$ and S, $\mathsf{Pre}(\mathfrak{A}_1) \in S$. So $\mathsf{Post}(\mathfrak{A}_1) \in S$.

Otherwise there is $\mathfrak{A}_{i-1} \in M$ such that $\mathsf{Post}(\mathfrak{A}_{i-1}) = \mathsf{Pre}(\mathfrak{A}_i)$, and by induction hypothesis $\mathsf{Post}(\mathfrak{A}_{i-1}) \in S$, so $\mathsf{Pre}(\mathfrak{A}_i) \in S$, and $\mathsf{Post}(\mathfrak{A}_i) \in S$.

By definition of $\rho_{\mathfrak{S}}(N)$, it suffices to show $\mathfrak{A}_i \in N$ for all $i > 0$, $\mathfrak{A}_i \in r$. Take any such $\mathfrak{A}_i$. By assumption $\mathfrak{A}_i \in M$ and by the above $\mathsf{Pre}(\mathfrak{A}_i) \in S$. So $\mathfrak{A}_i \in N$. □

3.4.3. d-validity

Definition (d-validity)**.** For $d \geqslant 0$, a formula ψ is d-*valid* whenever $\mathfrak{A}, \mu \vDash \psi$ for all $\mathfrak{A}, \mu : \langle\psi\rangle \to \mathrm{dom}(\mathfrak{A})$ such that for some $a, b \in \mathrm{Im}(\mu)$, $\delta_{\mathfrak{A}}(a, b) > d$. ■

This technical requirement of d-validity is related to the proviso that $G_{\mathcal{C}}$ is connected in a great clause $\mathcal{C}$. The lemmas below show its usefulness: given a match $m : p \to \mathfrak{B}$, a requirement of both context-freeness and retractability is that for some I, we must have $\mathfrak{B} \vDash \forall(I \vee m)$. During a run, we will see that $\mathsf{Pre}(\mathfrak{B}) \vDash \forall I$ is given by the previous run step (or by the initial state of the run). However, some elements may *appear*, i.e. not be in $\mathsf{Pre}(\mathfrak{B})$. For those, we do know that they have an m-preimage in p, so they are mostly taken care of by the "m" disjunct of $\mathfrak{B} \vDash \forall(I \vee m)$.

However there is a technical corner case, which we now discuss informally: suppose I has 2 free variables, $a, b \in \mathfrak{B}$, $a \notin \mathsf{Pre}(\mathfrak{B})$, and b does not have an m-preimage. So the "m" disjunct cannot help us. We need $\mathfrak{B} \vDash I(a, b)$, but that is not given to us by $\mathsf{Pre}(\mathfrak{B}) \vDash \forall I$ since $a \notin \mathsf{Pre}(\mathfrak{B})$. We will see in the lemma below that by their characterisation, a and b must be far apart in $\mathfrak{B}$, in fact, far enough apart that d-validity ensures $\mathfrak{B} \vDash I(a, b)$!

Lemma 3.4.3. If $\mathfrak{A}, \mathfrak{B} : \Theta$, I is d-valid, $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}(\llbracket\Delta\rrbracket_{\mathfrak{A}}, d)) \to \mathfrak{B}$, $\mathsf{Pre}(\mathfrak{B}) \in \mathsf{Init}(I)$, and $\mathfrak{B}$ is supported, then $\mathfrak{B} \vDash \forall(I \vee m)$

Proof. Take any $\nu : \langle I\rangle \to \mathrm{dom}(\mathfrak{B})$. If $\mathrm{Im}(\nu) \subseteq \mathrm{dom}(\mathsf{Pre}(\mathfrak{B}))$, we are done since $\mathrm{Pre}(\mathfrak{B}) \in \mathsf{Init}(I)$.

Otherwise, there is $a \in \mathrm{Im}(\nu) \setminus \mathrm{dom}(\mathsf{Pre}(\mathfrak{B}))$, so $\delta_{\mathfrak{B}}(a, \llbracket\neg\mathsf{Has}\rrbracket_{\mathfrak{B}}) = 0$.

Since $\mathfrak{B}$ is supported, $\llbracket\neg\mathsf{Has}\rrbracket_{\mathfrak{B}} \subseteq \llbracket\Delta\rrbracket_{\mathfrak{B}}$, so $\delta_{\mathfrak{B}}(a, \llbracket\Delta\rrbracket_{\mathfrak{B}}) = 0$. We get the following inclusions by the lemmas indicated below:

$$\mathcal{B}_{\mathfrak{B}}(\llbracket\Delta\rrbracket_{\mathfrak{B}}, d) \underset{\text{lemma 3.3.13}}{\subseteq} \mathcal{B}_{\mathfrak{B}}(m(\llbracket\Delta\rrbracket_{\mathfrak{A}}), d) \underset{\text{lemma 2.4.5}}{\subseteq} m(\mathfrak{A})$$

I is d-valid so wlog $\delta_{\mathfrak{B}}(a, b) \leqslant d$ for every $a, b \in \mathrm{Im}(\nu)$. We have $\mathrm{Im}(\nu) \subseteq \mathcal{B}_{\mathfrak{B}}(\llbracket\Delta\rrbracket_{\mathfrak{B}}, d) \subseteq m(\mathfrak{A})$, so we are done. □

The next lemma is key to taking care of the corner case mentioned above. It explains why great clauses must have a connected graph.

Lemma 3.4.4. If $\Gamma \vDash_{\rho} \phi$ then $\bigwedge\Gamma$ is $\|\phi\|$-valid.

Proof. First, note that if a formula is d-valid, it is d'-valid for all $d' \geqslant d$. We proceed by induction on the derivation.

- LITERAL$_\rho$: Trivial.
- CLAUSE$_\rho$: $\mathcal{C}$ is great so $G_{\mathcal{C}}$ is connected; trivially, so is $G_{\mathcal{C}^\circ}$. By lemma 3.3.12, if $\mathfrak{A}, \mu \nvDash \mathcal{C}^\circ$ then $\delta_{\mathfrak{A}}(a, b) \leqslant d$ for all $a, b \in \mathrm{Im}(\mu)$; this is the contrapositive phrasing of $|\mathrm{Im}(\mu)|$-validity. Since $|\mathrm{Im}(\mu)| \leqslant |\langle\mathcal{C}^\circ\rangle| = \|\forall\mathcal{C}\|$, we are done.

- DISJ$_\rho$, $\forall$GUARD$_\rho$, $\exists$GUARD$_\rho$, CIRCUM$_\rho$: Direct application of induction hypothesis: $\bigwedge\Gamma$ is $\|\phi\|$-valid and the formula in the conclusion has quantifier rank at least that of ϕ.
- CONJ$_\rho$: We have a $\|\phi_i\|$-valid $\bigwedge\Gamma_i$ for $i \in \{1,2\}$. It is easy to see that $\bigwedge\Gamma_1, \Gamma_2$ is $\max(\|\phi_1\|, \|\phi_2\|)$-valid; this happens to be the quantifier rank of $\phi_1 \wedge \phi_2$.

□

3.4.4. Run equality modulo context-free, retractable, d-validity

In this subsection we state the main lemmas which lead to the theorem. The next lemma simply says that support plays nice with validity in image complement.

Lemma 3.4.5. If $m : (\mathfrak{A}, K) \to \mathfrak{B}$ and $\mathfrak{B}$ is supported then $\mathfrak{B} \vDash \forall(\mathsf{Has} \vee m)$.

Proof. Let $a \notin m(\mathfrak{A})$ so by lemma 3.3.13 $a \notin [\![\Delta]\!]_{\mathfrak{B}}$. Since $\mathfrak{B}$ is supported, $a \in [\![\mathsf{Has}]\!]_{\mathfrak{B}}$. □

The two lemmas after that are dual of each other and correspond to the reasoning outlined in the introduction of this chapter.

Lemma 3.4.6. If ϕ is $(I \wedge \mathsf{Has}, d)$-context-free, I is d-valid, $\phi \vDash \forall I^\star$, and $\mathfrak{S} \in \mathrm{Init}(I)$, then $\rho_{\mathfrak{S}}(\mathrm{Inst}(\mathcal{P}(\phi, d))) \subseteq \rho_{\mathfrak{S}}(\phi)$

Proof. By lemma 3.4.2 we need $\mathfrak{B} \in [\![\exists\phi]\!]$ and $\mathrm{Post}(\mathfrak{B}) \in \mathrm{Init}(I)$ for every supported $\mathfrak{B} \in \mathrm{Insts}(\mathcal{P}(\phi, d))$ such that $\mathrm{Pre}(\mathfrak{B}) \in \mathrm{Init}(I)$. Since $\phi \vDash \forall I^\star$, it will suffice to show $\mathfrak{B}, \nu \vDash \phi$ for some $\nu : \langle\phi\rangle \to \mathrm{dom}(\mathfrak{B})$; this will imply both $\mathfrak{B} \in [\![\exists\phi]\!]$ and $\mathrm{Post}(\mathfrak{B}) \in \mathrm{Init}(I)$.

By definition of $\mathcal{P}(\phi, d)$, there is $(\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d)) \in \mathcal{P}(\phi, d)$ and $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d)) \to \mathfrak{B}$ with $\mu : \langle\phi\rangle \to \mathrm{dom}(\mathfrak{A})$ and $\mathfrak{A}, \mu \vDash \phi$. We will show $\mathfrak{B}, m \circ \mu \vDash \phi$. Since ϕ is context-free modulo $I \wedge \mathsf{Has}$, it suffices to show $\mathfrak{B} \vDash \forall((I \wedge \mathsf{Has}) \vee m)$. We get $\mathfrak{B} \vDash \forall(\mathsf{Has} \vee m)$ by lemma 3.4.5. Since $\mathcal{B}_{\mathfrak{A}}([\![\Delta]\!]_{\mathfrak{A}}, d) \subseteq \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d)$, by lemma 2.4.4 $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}[\![\Delta]\!]_{\mathfrak{A}}(d)) \to \mathfrak{B}$, so we can use lemma 3.4.3 to get $\mathfrak{B} \vDash \forall(I \vee m)$.

□

Lemma 3.4.7. If ϕ is $(I \wedge \mathsf{Has}, d)$-retractable, I is d-valid, $\phi \vDash \forall I^\star$, and $\mathfrak{S} \in \mathrm{Init}(I)$ then $\rho_{\mathfrak{S}}(\phi) \subseteq \rho_{\mathfrak{S}}(\mathrm{Inst}(\mathcal{P}(\phi, d)))$.

Proof. By lemma 3.4.2 we need $\mathfrak{B} \in \mathrm{Inst}(\mathcal{P}(\phi, d))$ and $\mathrm{Post}(\mathfrak{B}) \in \mathrm{Init}(I)$ for every supported $\mathfrak{B} \in [\![\exists\phi]\!]$ such that $\mathrm{Pre}(\mathfrak{B}) \in \mathrm{Init}(I)$. Since $\phi \vDash \forall I^\star$, $\mathrm{Post}(\mathfrak{B}) \in \mathrm{Init}(I)$; so it suffices to show $\mathfrak{B} \in \mathrm{Inst}(\mathcal{P}(\phi, d))$.

By assumption there is $\mu : \langle\phi\rangle \to \mathrm{dom}(\mathfrak{B})$ such that $\mathfrak{B}, \mu \vDash \phi$. Let $\mathfrak{A} := \mathfrak{B} \upharpoonright \mathcal{B}_{\mathfrak{B}}^{\Delta,\mu}(d+1)$. We trivially have $\mathrm{id}_{\mathfrak{A}} : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d)) \to \mathfrak{B}$.

By lemma 2.4.4, $\mathrm{id}_{\mathfrak{A}} : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}([\![\Delta]\!]_{\mathfrak{A}}, d)) \to \mathfrak{B}$ so by lemma 3.4.3, $\mathfrak{B} \vDash \forall(I \vee \mathrm{id}_{\mathfrak{A}})$. By lemma 3.4.5, $\mathfrak{B} \vDash \forall(\mathsf{Has} \vee m)$. So $\mathfrak{B} \vDash \forall((I \wedge \mathsf{Has}) \vee m)$. Since ϕ is retractable with d a retraction radius, $\mathfrak{A}, \mu \vDash \phi$. So $(\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d)) \in \mathcal{P}(\phi, d)$, so $\mathfrak{B} \in \mathrm{Inst}(\mathcal{P}(\phi, d))$. □

The next lemma is key to maintaining the invariant I in (I, d)-context-freeness and retractability throughout runs. It explains why DISJ$_\rho$ is additive, why $\vDash_\rho$ does not admit weakening, why the context of $\forall$GUARD$_\rho$ and $\exists$GUARD$_\rho$ is empty, and why great clauses must have dual, pre-post literals when they mention at least one literal with a symbol in **Dyn**.

Lemma 3.4.8. If $\Gamma \vDash_\rho \phi$ then $\phi \vDash \forall(\bigwedge\Gamma)^\star$

Proof. By induction on the derivation.

- LITERAL_ρ, DISJ_ρ, CONJ_ρ, CIRCUM_ρ, $\forall\text{GUARD}_\rho$,$\exists\text{GUARD}_\rho$: Trivial.
- CLAUSE_ρ $\mathcal{C}$ is great, so either $\mathcal{L}_{\text{Dyn}}(\mathcal{C})$ is empty, in which case $\mathcal{C} = \mathcal{C}^\star$, or it is not, in which case $\mathcal{C}^\star \equiv \top$.

$\square$

We can now say when $\rho_{\mathfrak{S}}(\psi) = \rho_{\mathfrak{S}}(\text{Insts}(\mathcal{P}(\psi, \|\psi\|)))$: when it is the conjunction of a $\vdash_\rho$-provable formula and *Support*, and when $\mathfrak{S}$ is nice enough:

Theorem 3.4.9. Let $\psi := \phi \wedge \mathit{Support}$. If $\Gamma \vdash_\rho \phi$ and $\mathfrak{S} \in \text{Init}(\bigwedge\Gamma)$ then

$$\rho_{\mathfrak{S}}(\psi) = \rho_{\mathfrak{S}}(\text{Inst}(\mathcal{P}(\psi, \|\psi\|)))$$

Proof. To apply lemmas 3.4.6 and 3.4.7 we need $\phi \wedge \mathit{Support}$ to be $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\phi \wedge \mathit{Support}\|)$-context-free and retractable, $\bigwedge\Gamma$ to be $\|\phi \wedge \mathit{Support}\|$-valid, and $\phi \wedge \mathit{Support} \vDash \forall(\bigwedge\Gamma)^\star$.

By lemma 3.3.22, ϕ is $(\bigwedge\Gamma \wedge \mathsf{Has}, \|\phi\|)$-context-free and retractable. By lemma 3.3.16, $\phi \wedge \mathit{Support}$ is as well. By lemma 3.4.4, $\bigwedge\Gamma$ is $\|\phi\|$-valid, so it is $\|\phi \wedge \mathit{Support}\|$-valid. By lemma 3.4.8, $\phi \vDash \forall(\bigwedge\Gamma)^\star$, so $\phi \wedge \mathit{Support} \vDash \forall(\bigwedge\Gamma)^\star$.

$\square$

3.5. Main theorem

This section summarises the last two chapters by saying when the runs of a formula are equal to the execution of the rules induced by the canonical patterns of that formula. It is simply a matter of applying theorems 3.4.9 and 2.6.4.

Theorem 3.5.1. If $\Gamma \vdash_\rho \phi$ and $\mathfrak{M} \in \text{Init}(\bigwedge\Gamma \wedge \mathsf{Has})$, then

$$\rho_{\mathfrak{M}}(\phi_{S0}) = \mathtt{exec}_{\mathfrak{M}}(\mathtt{rules}(\mathcal{P}(\phi_{S0}, \|\phi_{S0}\|)))$$

Proof. Note that $\mathit{Support}_0 = \mathit{Support} \wedge \mathsf{Has}_0 \wedge \mathsf{Has}_0^\star$. By lemma A.0.1 (given in appendix), there is $\psi \equiv \mathsf{Has}_0 \wedge \mathsf{Has}_0^\star$ and Γ' such that $\Gamma' \vdash_\rho \psi$, so $\Gamma, \Gamma' \vdash_\rho \phi \wedge \psi$. Since $\mathfrak{M} \in \text{Init}(\bigwedge\Gamma)$, theorem 3.4.9 is applicable to $\phi \wedge \psi$, so

$$\rho_{\mathfrak{M}}(\phi_{S0}) = \rho_{\mathfrak{M}}(\text{Inst}(\mathcal{P}(\phi_{S0}, \|\phi_{S0}\|)))$$

By definition any pattern in $\mathcal{P}(\phi_{S0}, \|\phi_{S0}\|)$ is pure and strongly supported. $\mathfrak{M} \in \text{Init}(\mathsf{Has})$, so theorem 2.6.4 is applicable, so

$$\rho_{\mathfrak{M}}(\text{Inst}(\mathcal{P}(\phi_{S0}, \|\phi_{S0}\|))) = \mathtt{exec}_{\mathfrak{M}}(\mathtt{rules}(\mathcal{P}(\phi_{S0}, \|\phi_{S0}\|)))$$

$\square$

We complete figure 2.9 and show in figure 3.9 the connection between the runs on the instances of $\mathcal{P}(\phi, \|\phi\|)$ and the runs on the models of ϕ. The satisfaction relation on the right is missing its interpretation component.

$$\begin{tikzcd}
(\mathfrak{A}, K) \arrow[d] & (\mathfrak{A}', K') \arrow[d] & (\mathfrak{A}'', K'') \arrow[d] & \text{canonical patterns of} & \phi \arrow[d, phantom, "\rotatebox{90}{$\models$}"] & \phi \arrow[d, phantom, "\rotatebox{90}{$\models$}"] & \phi \arrow[d, phantom, "\rotatebox{90}{$\models$}"] \\
\langle \mathfrak{M}_0, \mathfrak{M}_1 \rangle & \langle \mathfrak{M}_1, \mathfrak{M}_2 \rangle & \langle \mathfrak{M}_2, \mathfrak{M}_3 \rangle & = & \langle \mathfrak{M}_0, \mathfrak{M}_1 \rangle & \langle \mathfrak{M}_1, \mathfrak{M}_2 \rangle & \langle \mathfrak{M}_2, \mathfrak{M}_3 \rangle
\end{tikzcd}$$

Figure 3.9.: Left: a run in $\rho_{\mathfrak{M}_0}(\mathrm{Insts}(P))$. Right: a run in $\rho_{\mathfrak{M}_0}(\phi)$

4. Bounded change-minimal specifications

In the previous chapter we showed when runs on formulas can be interpreted as the execution of a set of rules. But is that set of rules finite? In this chapter, we show when that set of rules can be finite. This will be done by defining another set of deduction rules (the name clash is unfortunate), $\vdash_{\lhd}$, such that $\vdash_{\lhd}$-provable formulas have a finite set of canonical patterns.

The proof uses a property called preservation. Essentially, models of a preserved formula ϕ can lose changes in a very controlled way and still satisfy ϕ. Preservation is useful as a property because it plays well with logical operations.

A first step is to restrict the class of transitions further. We introduce *Bounded Forests with Links* in section 4.1. One crucial property of BFLs is that the graph of functions in $\mathbf{Stat}_{\mathsf{fun}}$ is bounded by some parameter n. In section 4.2 we define preservation. First, we introduce a parameterized operation on BFLs called *taking a sub*. A formula ϕ is preserved (modulo some parameters) if the sub of a model of ϕ is still a model of ϕ. In section 4.3 we show why preservation leads to a finite set of canonical patterns, and finally in section 4.4 we introduce deduction rules to build preserved formulas.

We start by going back to the issue of an infinite set of rules. If $\mathcal{P}(\phi, \|\phi\|)$ is infinite, theorem 3.5.1 is not very useful. If there are infinitely many patterns to sift through, an execution engine for $\rightharpoonup$ cannot do much good. Where does the infinity come from? First, there is no bound on the size of a ball (even of radius 0) around a element of a transition. We will restrict the class of transitions so that for every element, its d-ball is bounded for all $d \geqslant 0$.

Moreover, the domains and application conditions of the canonical patterns are of the form $\mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d)$, i.e. balls around two sets: 1) the image of an interpretation (μ), and 2) changes (Δ). The image of μ is bounded by the formula, but the number of changes is not. The min operator $\downarrow$ can help, but there is still work to do: a formula may very well be of the form $\downarrow\phi$ yet have infinitely many canonical patterns.

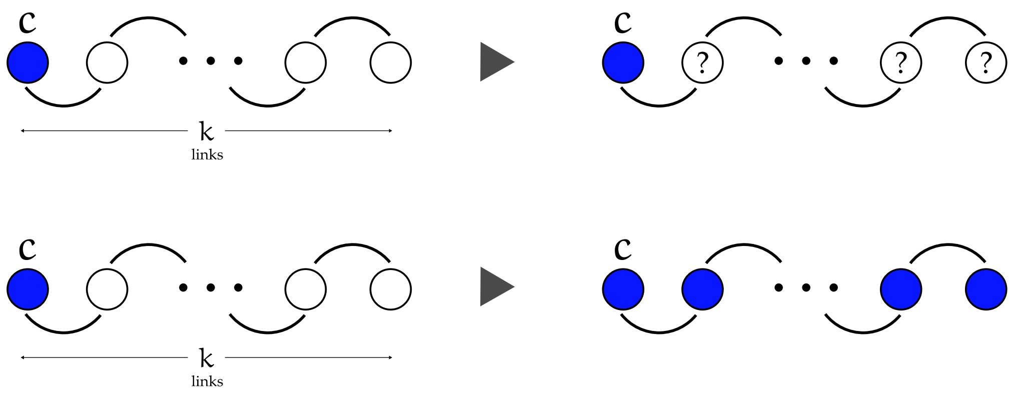


Figure 4.1.: Precondition of $\mathfrak{A}_k$. $\mathfrak{A}_k \vDash \downarrow CC$, the entire connected component of c must become Blue.

Example. Suppose $\mathsf{Link}, \mathsf{Link}^\star$ or of outdegree 2 at most. Consider:

$$\begin{aligned}\downarrow CC \coloneqq \downarrow \big(&\mathsf{Blue}(c) \wedge \\ &\forall x.\ (x \neq c \rightarrow \neg \mathsf{Blue}(x)) \wedge \\ &\forall y.\ \mathsf{Blue}^\star(x) \rightarrow \mathsf{Link}^\star(x, y) \rightarrow \mathsf{Blue}^\star(y)\big)\end{aligned}$$

with $\mathsf{Blue} \in \mathbf{Dyn}$. $CC(c)$ specifies *a)* that the interpretation of c is Blue and no other element is blue, and *b)* an element with a $\mathsf{Blue}^\star$ neighbor must be $\mathsf{Blue}^\star$.

For every $k \geqslant 0$ consider the transition $\mathfrak{A}_k$, with a precondition as in figure 4.1. There is a connected component (for the relation Link) of size k. Since $\mathsf{Blue}^\star$ is true for $[\![c]\!]_{\mathfrak{A}_k}$, $\mathsf{Blue}^\star$ must be true of $[\![c]\!]_{\mathfrak{A}_k}$'s neighbour, and so on for the entire connected component. For every $k \geqslant 0$, the transition of figure 4.1 is a model of $\downarrow CC$. So the number of canonical patterns for $\downarrow CC$ is infinite even up to isomorphism. ■

4.1. Bounded Forests with Links

In this section we define bounded forests with links and show the theories that characterise them. The proofs are in the appendix.

We are moving to a more restricted class of transitions, so that a transition can always be described as a pair of forests with trees of bounded height, linked by a functional and symmetric relation. The bounded height of the trees is necessary to get a finite number of canonical patterns.

Definition (BFL signature)**.** Θ is a *BFL signature* if $\mathbf{Stat}_{\mathrm{fun}}$ contains a distinguished function symbol parent. ■

Definition (n-BFL)**.** If $n \geqslant 0$, Θ is a BFL signature and $\mathfrak{A} : \Theta$, $\mathfrak{A}$ is an *n-BFL* (notation: $\mathfrak{A} : \Theta_n$) whenever:

- The loop-free union of the graphs of functions $[\![f]\!]_{\mathfrak{A}}$ for $f \in \mathbf{Stat}_{\mathrm{fun}} \setminus \{\mathrm{parent}\}$ is a forest.
- The trees of that forest have height at most n.
- The loop-free graph of $[\![\mathrm{parent}]\!]_{\mathfrak{A}}$ is the parent relation in that forest.
- The relations $[\![\mathsf{Link}]\!]_{\mathfrak{A}}$ and $[\![\mathsf{Link}^\star]\!]_{\mathfrak{A}}$ are symmetric and functional.

■

The parent function makes the proofs simpler since any embedding is now a 0-embedding (since accessibility through the function graph is now symmetric).

Note that if $\mathfrak{A} : \Theta_n$, the restriction of $G_{\mathfrak{A}}$ to 0-edges, $G^0_{\mathfrak{A}}$, is the undirected version of the forest in the definition of n-BFL.

Implicit parameter [BFL height] $n \geqslant 0$ is an arbitrary maximum height.

Definition (Tree)**.** A tree of t of $G_{\mathfrak{A}}$ is the restriction of $G^0_{\mathfrak{A}}$ to one of its connected components. $V_t \subseteq \mathrm{dom}(\mathfrak{A})$ is the vertex set of that tree. If $a \in \mathrm{dom}(\mathfrak{A})$, t_a is the tree of $G_{\mathfrak{A}}$ that contains a. ■

It is easy to see that $V_{t_a} = \mathcal{B}_{\mathfrak{A}}(a, 0)$ for all $a \in \mathrm{dom}(\mathfrak{A})$ and that for any $b \in V_{t_a}$, $b \in \mathfrak{A} \restriction \{a\}$.

Example. In figure 4.2, the transition $\mathfrak{A}$ has two trees. We have $\mathbf{Stat}_{\mathrm{fun}} = \{f, g, \mathrm{parent}\}$, and $\mathfrak{A}$ is a 1-BFL. The interpretations of Link and $\mathsf{Link}^\star$ are symmetric and functional (no change from earlier graphical representations). The parent function is indeed the inverse of all other functions everywhere except on tree leafs. The tree containing a is t_a, and $V_{t_a} = \mathcal{B}_{\mathfrak{A}}(\{a\}, 0) = \{a, b, c\}$. ■

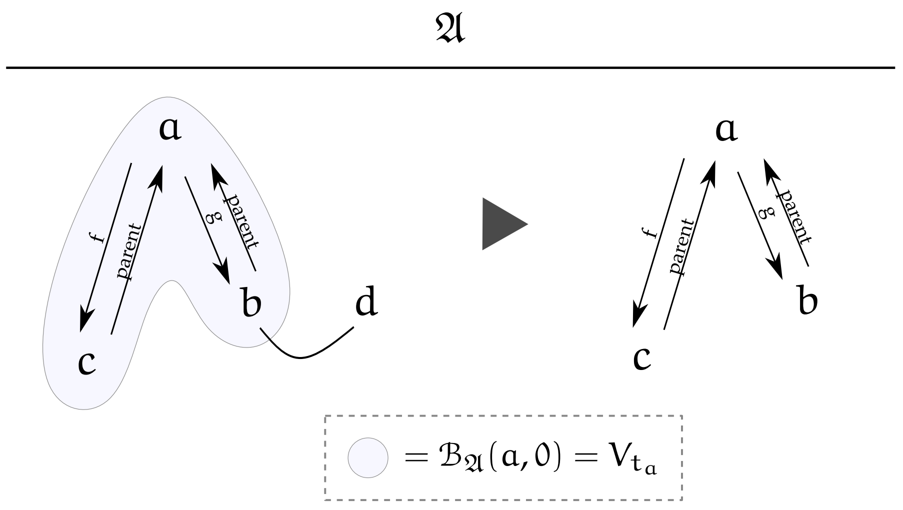


Figure 4.2.: A 1-BFL $\mathfrak{A}$

It is possible to characterise n-BFLs with universal, first-order formulas:

Definition. The theory of n-BFLs is:

$$\mathcal{T}_n := \mathit{Symlink} \wedge \mathit{Funlink} \wedge \mathcal{T}'_n$$

where

$$\begin{aligned}
\mathcal{T}'_n &:= \forall x. \mathit{PSpec}(x) \wedge H^n(x) \\
\mathit{PSpec}(x) &:= \Big(\bigwedge_{f \in \mathbf{Stat}_{\mathrm{fun}}} f(x) \neq x \to \mathrm{parent}(f(x)) = x \Big) \\
&\quad \wedge \Big(\mathrm{parent}(x) \neq x \to \bigvee_{f \in \mathbf{Stat}_{\mathrm{fun}}} f(\mathrm{parent}(x)) = x \Big) \\
H^i(x) &:= \bigwedge_{f \in \mathbf{Stat}_{\mathrm{fun}}} f(x) \neq x \to H^{i-1}(f(x)) && \text{(for } 0 \leqslant i < n\text{)} \\
H^{-1}(x) &:= \bot \\
\mathit{FunLink}' &:= \forall x, y, z.\ \mathsf{Link}(x, y) \to \mathsf{Link}(x, z) \to y = z \\
\mathit{FunLink} &:= \mathit{FunLink} \wedge \mathit{FunLink}^\star \\
\mathit{Symlink}' &:= \forall x, y.\ \mathsf{Link}(x, y) \to \mathsf{Link}(y, x) \\
\mathit{Symlink} &:= \mathit{Symlink} \wedge \mathit{Symlink}^\star
\end{aligned}$$

■

Lemma 4.1.1. If $\mathfrak{A} : \Theta$, $\mathfrak{A} : \Theta_n$ iff $\mathfrak{A} \vDash \mathcal{T}_n$.

See proof in Appendix A.

Definition. $\mathcal{T}_{ns} := \mathcal{T}_n \wedge \mathit{Support}$ ■

4.2. Preservation

In this section we define preservation and give some examples to help build intuition. The definition is a bit involved and goes through intermediary steps. First, some elements in an n-BFL are considered active if they are involved in a particular kind of change. Next, a tree t in an n-BFL $\mathfrak{A}$ can be *cleared*, producing a *sub*transition $\mathfrak{A}'$. This clearing operation removes changes involving t in a very controlled way. For instance, it does not remove changes that touch trees with active elements. Subtransitions are parameterised by a set of elements and a distance. We then define preservation on formulas by saying that for a formula ϕ, if satisfaction is closed under the operation of clearing a tree (thus producing a sub) modulo those parameters, then ϕ is preserved modulo those parameters.

We conclude with more examples to help understand subs and preservation.

Notation (Useful formula macros)**.** If $A \in \mathbf{Dyn}$ and $\mathbf{x}$ is a tuple of variables of size the arity of A, then:

$$\oplus A(\mathbf{x}) := \neg A(\mathbf{x}) \wedge A^\star(\mathbf{x}) \qquad \text{(“}A(\mathbf{x})\text{ becomes true”)}$$
$$\ominus A(\mathbf{x}) := A(\mathbf{x}) \wedge \neg A^\star(\mathbf{x}) \qquad \text{(“}A(\mathbf{x})\text{ becomes false”)}$$

■

Definition (Operations on relations and sets)**.** If R is a relation and X is a set,

$$R \mathbin{/} X := \{\, \mathbf{r} \in R \mid \mathbf{r} \cap X \neq \emptyset \,\} \qquad \text{(tuples that mention } X\text{)}$$
$$R \setminus X := R \setminus (R \mathbin{/} X) \qquad \text{(tuples that do not mention } X\text{)}$$

■

For instance, $\{(a,b),(a,c)\} \mathbin{/} \{b\} = \{(a,b)\}$ and $\{(a,b),(a,c)\} \setminus \{b\} = \{(a,c)\}$. The operation $\setminus$ generalizes the usual difference in that the definition reduces to set or relation difference one when R is a unary relation and either R is understood as a set, or when X is understood as a set of 1-tuples.

We move on to preservation. First, some preliminary definitions:

Definition (Active elements and trees)**.** If $\mathfrak{A} : \Theta_n$, $a \in \mathfrak{A}$ is *active* whenever at least one of the following is true:

- $a \in [\![\Delta A]\!]_{\mathfrak{A}}$, for some $A \in \mathbf{Dyn}$ unary
- There is $b \in V_{t_a}$ such that $(a,b) \in [\![\ominus \mathsf{Link}]\!]_{\mathfrak{A}}$
- There is $b \in \mathrm{dom}(\mathfrak{A})$ such that $(a,b) \in [\![\oplus \mathsf{Link}]\!]_{\mathfrak{A}}$

A tree t of $G_{\mathfrak{A}}$ is active if it contains at least one active element. The set of elements in active trees is $\mathcal{A}(\mathfrak{A})$, and for t a tree of $G_{\mathfrak{A}}$, the set of elements in active trees different from t is $\mathcal{A}_{\bar{t}}(\mathfrak{A}) := \mathcal{A}(\mathfrak{A}) \setminus V_t$. ■

In plain english, a elements is active if it is involved in any change other than an edge deletion involving a different tree.

Example. In figure 4.3, the transition $\mathfrak{A}$ contains 2 trees, one with the elements a, b, c and one with the single element d. The active elements are in red. a is active because it is in $[\![\Delta A]\!]_{\mathfrak{A}}$, and c is active because it loses an internal edge. However, b and d both lose an edge connected to another tree (the orientation of the edge is irrelevant), so they are not active. As a result, t_a is active because it has at least one active element, but t_d is not active. ■

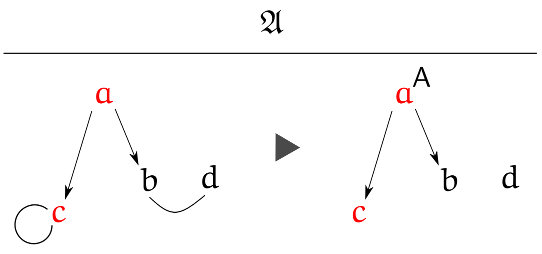


Figure 4.3.: active elements in red.

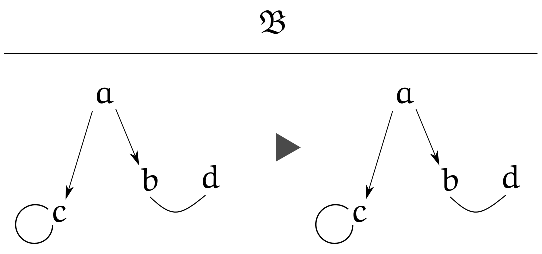


Figure 4.4.: $\mathfrak{B}$, after clearing t_a in $\mathfrak{A}$.

Definition (Subs)**.** If $\mathfrak{A} : \Theta_n$, $\mathfrak{B} \trianglelefteq \mathfrak{A}$, $K \subseteq \mathrm{dom}(\mathfrak{B})$, $d \geqslant 0$ and t is a tree of $G_{\mathfrak{A}}$, $\mathfrak{B}$ is a (K, d)-*sub* of $\mathfrak{A}$ with cleared tree t whenever there is $K' \subseteq \mathrm{dom}(\mathfrak{B})$ such that:

$$
\begin{aligned}
V_t \cap K' &= \emptyset \\
\mathcal{A}_{\bar{t}} \cup \mathcal{B}_{\mathfrak{A}}(K, d) &\subseteq K' \\
[\![\Delta A]\!]_{\mathfrak{B}} &= [\![\Delta A]\!]_{\mathfrak{A}} \setminus V_t && \text{(for all unary } A \in \textbf{Dyn}\text{)} \\
[\![\oplus \mathsf{Link}]\!]_{\mathfrak{B}} &= [\![\oplus \mathsf{Link}]\!]_{\mathfrak{A}} \setminus V_t \\
[\![\ominus \mathsf{Link}]\!]_{\mathfrak{B}} &= ([\![\ominus \mathsf{Link}]\!]_{\mathfrak{A}} \setminus V_t) \cup ([\![\ominus \mathsf{Link}]\!]_{\mathfrak{A}} / \mathcal{B}_{\mathfrak{B}}(K', 0))
\end{aligned}
$$

If t is not specified, we say that $\mathfrak{B}$ is *a* (K, d)-sub of $\mathfrak{A}$, and if K, d are not specified, we say that $\mathfrak{B}$ is a sub of $\mathfrak{A}$. ■

In plain english, in a (K, d)-sub with cleared tree t, we

1. Pick a 0-ball around a protected set. It must contain at least a certain ball and active elements outside of t

2. Make sure t does not touch the protected set

3. Make t inactive

4. Also remove edge deletions that involve t and any edge deletion with one leg in t and the other unprotected.

5. Possibly remove any unprotected element of $\mathrm{dom}(\mathfrak{A})$ in a way compatible with $\trianglelefteq$.

The idea is that gradually removing changes this way lets us keep satisfying a class of formulas on the way down to minimal changes.

If a change (unary predicate change, or edge deletion, or edge addition) is present in $\mathfrak{A}$ but not present in $\mathfrak{B}$, we say that the change has been *cleared*.

Example. Clearing t_a in $\mathfrak{A}$ from figure 4.3 (and picking a minimal K') result in $\mathfrak{B}$, a $(\emptyset, 0)$-sub of $\mathfrak{A}$, seen figure 4.4. All changes from t_a are removed, *including the edge deletion between* b *and* d. The edge deletion is cleared because t_d is not active.

Contrast with transitions $\mathfrak{A}'$ (figure 4.5) and $\mathfrak{B}'$ (figure 4.6). The difference between $\mathfrak{A}$ and $\mathfrak{A}'$ is that d is active in $\mathfrak{A}'$ (because $d \in [\![\Delta A]\!]_{\mathfrak{A}'}$). Just as $\mathfrak{B}$ is a $(\emptyset, 0)$-sub of $\mathfrak{A}$ with cleared tree t_a, $\mathfrak{B}'$ is a $(\emptyset, 0)$-sub of $\mathfrak{A}'$ with cleared tree t_a. Note the difference: since t_d is active in $\mathfrak{A}'$, the link deletion between b and d *has not been cleared*. However, clearing t_d in $\mathfrak{B}'$ would finally clear that link deletion. ■

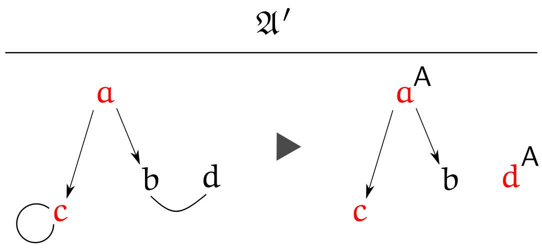


Figure 4.5.: $\mathfrak{A}$ with an active d.

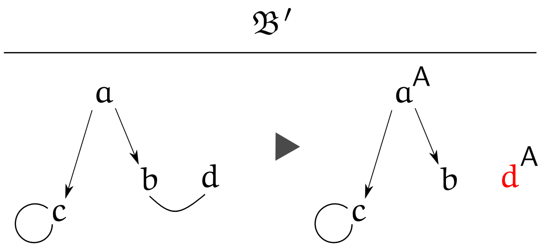


Figure 4.6.: $\mathfrak{B}'$, after clearing t_a in $\mathfrak{A}'$.

Definition (Preservation)**.** If $d \geqslant 0$ and $\mathcal{V}$ is a set of variables a formula ϕ is *$(\mathcal{V}, d)$-preserved* whenever for all $\mathfrak{A} : \Theta_n$, $\mu : \mathcal{V} \to \mathrm{dom}(\mathfrak{A})$, $(\mu(\mathcal{V}), d)$-sub $\mathfrak{B}$ of $\mathfrak{A}$ and $\nu : \langle\phi\rangle \setminus \mathcal{V} \to \mathrm{dom}(\mathfrak{B})$, if $\mathfrak{A}, \mu, \nu \models \phi$ then $\mathfrak{B}, \mu, \nu \models \phi$.

■

Informally, ϕ is $(\mathcal{V}, d)$-preserved if ϕ-satisfaction is closed under $(\mathcal{V}, d)$-sub.

Examples.

- Consider the formula $A^\star(x)$. If $a \in [\![A^\star]\!]_{\mathfrak{A}} \setminus [\![A]\!]_{\mathfrak{A}}$, that remains true for any sub of a as long as a is not cleared. So $A^\star(x)$ is $(\{x\}, 0)$-preserved. In general, postcondition constraints increase the level of protection necessary for a formula to remain preserved.
- Consider $\forall x.\ A(x)$. Since only the precondition is mentioned, taking a sub cannot break satisfaction of the formula. In general, precondition constraints cannot impact preservation.
- Consider $\mathsf{Link}^\star(x, y)$. If a link "appears" between the interpretation of x and y and either of them is cleared, the link creation will be cleared as well. So they must both be protected: the formula is $(\{x, y\}, 0)$-preserved, as well as $(\{x\}, 1)$-preserved and $(\{y\}, 1)$-preserved. In general, positive link constraints should be protected on both sides.
- Consider $\exists x.\ N(x)$, with $N \in$ **Stat** a predicate symbol. With the model $\mathfrak{A}$ with $\{a, b\} = \mathrm{dom}(\mathfrak{A})$ and $\{a\} = [\![N]\!]_{\mathfrak{A}}$, removing a yields a $(\emptyset, 0)$-sub $\mathfrak{B}$ which is not a model of $\exists x.\ N(x)$. In general, without precautions, existential quantification can easily make formulas be not-preserved, even if the formula below the $\exists$ does not talk about dynamic properties at all.

■

4.3. Preservation leads to finitely many canonical patterns

In this section, we show that preserved formula of a certain shape have finitely many canonical patterns.

Overall, preservation leads to finitely many canonical patterns (for minimised formulas) because as one takes subs, one goes down the change order $\trianglelefteq$. Eventually all changes are inside a protected area defined as a ball around the interpretation of variables. Therefore there is a bounded number of changes in the minimal models of a preserved formula. Since function graphs in n-BFLs are bounded, balls around elements have bounded size; and therefore corresponding canonical patterns have bounded size.

The next two lemmas are weakening lemmas:

Lemma 4.3.1. If $\mathfrak{B}$ is a (K, d)-sub of $\mathfrak{A}$ and $\mathcal{B}_{\mathfrak{A}}(K', d') \subseteq \mathcal{B}_{\mathfrak{A}}(K, d)$ then $\mathfrak{B}$ is a (K', d')-sub of $\mathfrak{A}$.

Proof. Trivial. □

Lemma 4.3.2. If ϕ is $(\mathcal{V}, d)$-preserved, $\mathcal{V} \subseteq \mathcal{V}'$ and $d \leqslant d'$, ϕ is $(\mathcal{V}', d)$-preserved.

Proof. Let X be the variables of $\mathcal{V}$, Y be the variables of $\mathcal{V}' \setminus \mathcal{V}$.

Let $\mathfrak{A} : \Theta_n$. Let $\mu_Y : X \uplus Y \to \mathrm{dom}(\mathfrak{A})$ be any interpretation, and $\mu : X \to \mathrm{dom}(\mathfrak{A}) := \mu_Y \restriction X$. Let $\mathfrak{B}$ be a $(\mathrm{Im}(\mu_Y), d)$-sub of $\mathfrak{A}$, $\nu_Y : \langle\phi\rangle \setminus (X \uplus Y) \to \mathrm{dom}(\mathfrak{B})$ be any interpretation, and $\nu : \langle\phi\rangle \setminus Y \to \mathrm{dom}(\mathfrak{B}) := \nu_Y \uplus (\mu_Y \restriction Y)$. We have $\mu_Y \uplus \nu_Y = \mu \uplus \nu$.

By lemma 4.3.1 and assumption on ϕ, $\mathfrak{B}, \mu, \nu \models \phi$. So $\mathfrak{B}, \mu_Y, \nu_Y \models \phi$. □

Here we see that taking subs does not break out of the class of n-BFLs:

Lemma 4.3.3. If $\mathfrak{A} : \Theta_n$ and $\mathfrak{B}$ is a sub of $\mathfrak{A}$, then $\mathfrak{B} : \Theta_n$.

Proof. First, we show that Link and $\mathsf{Link}^\star$ are symmetric and functional.

By definition of $\trianglelefteq$, $[\![\mathsf{Link}]\!]_{\mathfrak{B}} = [\![\mathsf{Link}]\!]_{\mathfrak{A}}$, so $[\![\mathsf{Link}]\!]_{\mathfrak{B}}$ is symmetric and functional.

So we need $[\![\Delta\mathsf{Link}]\!]_{\mathfrak{B}}$ symmetric and functional. It is defined by a restriction of $[\![\Delta\mathsf{Link}]\!]_{\mathfrak{A}}$, which is symmetric and functional, to subrelations of $[\![\mathsf{Link}]\!]_{\mathfrak{A}}$ and $[\![\mathsf{Link}^\star]\!]_{\mathfrak{A}}$, which are symmetric and functional. These subrelations are defined by restriction to sets or deletion of sets, so they are symmetric and functional as well. Simple set-theoretic considerations yield the result.

Now we want the loop-free union of $[\![h]\!]_{\mathfrak{B}}$ for $h \in \mathbf{Stat}_{\mathrm{fun}} \setminus \{\mathrm{parent}\}$ to be a tree and $[\![\mathrm{parent}]\!]_{\mathfrak{B}}$ to be the parent function of that tree. By definition of $\trianglelefteq$, $\mathrm{dom}(\mathfrak{B}) \subseteq \mathrm{dom}(\mathfrak{A})$ and $[\![\mathbf{Stat}]\!]_{\mathfrak{B}} = [\![\mathbf{Stat}]\!]_{\mathfrak{A}} \restriction \mathrm{dom}(\mathfrak{B})$, so it suffices to show that for $e \in V_{G_{\mathfrak{A}}}$, with t_e the tree of e in $G_{\mathfrak{A}}$, $e \notin \mathrm{dom}(\mathfrak{B})$ implies $V_{t_e} \cap \mathrm{dom}(\mathfrak{B}) = \emptyset$ (that is, that removing any element removes their entire tree).

If e has no parent or child in t_e, we are done. If there is v parent of e in $G_{\mathfrak{A}}$, there is $f \in \mathbf{f}$ such that $[\![f]\!]_{\mathfrak{A}}(v) = e$. So $v \in \mathrm{dom}(\mathfrak{B})$ implies $e \in \mathrm{dom}(\mathfrak{B})$. If there is v child of e, then $[\![\mathit{parent}]\!]_{\mathfrak{A}}(v) = e$, and again $v \in \mathrm{dom}(\mathfrak{B})$ implies $e \in \mathrm{dom}(\mathfrak{B})$. The proof goes by induction on the distance to e. □

Definition. Let

$$\mathit{Supp}_D := \forall xy.\ (\mathsf{Link}(x,y) \vee \mathsf{Link}(y,x) \vee \bigvee_{A \in \mathcal{D}yn \text{ unary}} A(x)) \to \mathsf{Has}(x)$$

$$\mathit{Support}_D := \mathit{Support} \wedge \mathit{Supp}_D \wedge \mathit{Supp}_D^\star$$

■

Definition. $\mathcal{T}_{ns_D} := \mathcal{T}_n \wedge \mathit{Support}_D$ ■

This strengthening of support requires that an element with at least some true dynamic property is deemed to be present.

A transition is *dynamically supported* when it satisfies $\mathit{Support}_D$, and a formula is *dynamically supported* when it implies $\mathit{Support}_D$. $\mathit{Support}_D$ is this chapter's version of $\mathit{Support}_0$: a formula stronger than *Support* necessary for some inductions to go through or preservation properties to hold.

The following lemma shows that by repeatedly taking subs from a dynamically supported transition, we reach a point where all changes are within a small boundary:

Lemma 4.3.4. If $\mathfrak{A} : \Theta_n$ is dynamically supported, $d \geqslant 0$, $K \subseteq \mathrm{dom}(\mathfrak{A})$, there exists a dynamically supported (K, d)-sub $\mathfrak{B}$ of $\mathfrak{A}$ with $\mathfrak{B} \neq \mathfrak{A}$ whenever either

- There is $a \notin \mathcal{B}_{\mathfrak{A}}(K, d)$ such that $a \in [\![\Delta A]\!]_{\mathfrak{A}}$ for some $A \in \mathbf{Dyn}$, or such that $(a, b) \in [\![\oplus \mathsf{Link}]\!]_{\mathfrak{A}}$ for some b.
- There is $a \notin \mathcal{B}_{\mathfrak{A}}(K, d+1)$ such that $(a, b) \in [\![\ominus \mathsf{Link}]\!]_{\mathfrak{A}}$ for some b.

Proof. We start by defining the cleared tree t. If t_a (the tree of $G_{\mathfrak{A}}$ that contains a) is active, let $t := t_a$. Otherwise, by definition of active, there is $b \in \mathrm{dom}(\mathfrak{A})$ with tree $t_b \neq t_a$ and $(a, b) \in [\![\ominus \mathsf{Link}]\!]_{\mathfrak{A}}$. Let $t := t_b$.

Let $\mathfrak{B}$ be as follows, for $A \in \mathbf{Dyn}$ unary:

$$[\![\oplus \mathsf{Link}]\!]_{\mathfrak{B}} := [\![\oplus \mathsf{Link}]\!]_{\mathfrak{A}} \setminus V_t$$
$$[\![\ominus \mathsf{Link}]\!]_{\mathfrak{B}} := ([\![\ominus \mathsf{Link}]\!]_{\mathfrak{A}} \setminus V_t) \cup \Big([\![\ominus \mathsf{Link}]\!]_{\mathfrak{A}} \,/\, (\mathcal{A}_{\bar{t}}(\mathfrak{A}) \cup \mathcal{B}_{\mathfrak{A}}(K, d))\Big)$$
$$[\![\Delta A]\!]_{\mathfrak{B}} := [\![\Delta A]\!]_{\mathfrak{A}} \setminus V_t \qquad [\![\mathbf{Dyn}]\!]_{\mathfrak{B}} := [\![\mathbf{Dyn}]\!]_{\mathfrak{A}}$$

which induces $[\![A^\star]\!]_{\mathfrak{B}}$ for all $A \in \mathbf{Dyn}$. Now let $\mathrm{dom}(\mathfrak{B}) := [\![\mathsf{Has}]\!]_{\mathfrak{B}} \cup [\![\mathsf{Has}^\star]\!]_{\mathfrak{B}}$, and $[\![\mathbf{Stat}]\!]_{\mathfrak{B}} := [\![\mathbf{Stat}]\!]_{\mathfrak{A}} \restriction \mathrm{dom}(\mathfrak{B})$.

We show that $\mathfrak{B}$ well-defined: it is trivial for the static part; if $b \in \mathrm{dom}(\mathfrak{A})$ is in the underlying set of $[\![A]\!]_{\mathfrak{A}} = [\![A]\!]_{\mathfrak{B}}$ for some $A \in \mathbf{Dyn}$, since $\mathfrak{A}$ is dynamically supported, $b \in [\![\mathsf{Has}]\!]_{\mathfrak{B}} \subseteq \mathrm{dom}(\mathfrak{B})$. Otherwise, suppose b is in the underlying set of $[\![\oplus A]\!]_{\mathfrak{A}}$. Again by dynamic support, $b \in [\![\oplus \mathsf{Has}]\!]_{\mathfrak{A}}$. If $b \in V_t$ then $b \notin [\![\mathsf{Has}^\star]\!]_{\mathfrak{B}}$, so $b \notin \mathrm{dom}(\mathfrak{B})$, otherwise $b \in [\![\mathsf{Has}^\star]\!]_{\mathfrak{B}}$, so $b \in \mathrm{dom}(\mathfrak{B})$. The same reasoning shows that $\mathfrak{B}$ is dynamically supported.

We show that $\mathfrak{B} \trianglelefteq \mathfrak{A}$. $[\![\mathbf{Dyn}]\!]_{\mathfrak{B}} = [\![\mathbf{Dyn}]\!]_{\mathfrak{A}}$ and $[\![\mathbf{Stat}]\!]_{\mathfrak{B}} = [\![\mathbf{Stat}]\!]_{\mathfrak{A}} \restriction \mathrm{dom}(\mathfrak{B})$ by definition. Since for $A \in \mathbf{Dyn}$, $[\![\Delta A]\!]_{\mathfrak{B}}$ is defined by restriction, we have $[\![\Delta \mathbf{Dyn}]\!]_{\mathfrak{B}} \subseteq [\![\Delta \mathbf{Dyn}]\!]_{\mathfrak{A}}$. Since $[\![\mathsf{Has}]\!]_{\mathfrak{B}} = [\![\mathsf{Has}]\!]_{\mathfrak{A}}$ and $[\![\mathsf{Has}^\star]\!]_{\mathfrak{B}}$ is defined by restriction of $[\![\mathsf{Has}^\star]\!]_{\mathfrak{A}}$, $\mathrm{dom}(\mathfrak{B}) \subseteq \mathrm{dom}(\mathfrak{A})$.

By construction, $\mathfrak{B}$ is a (K, d)-sub of $\mathfrak{A}$ (with $K' = \mathcal{A}_{\bar{t}} \cup \mathcal{B}_{\mathfrak{A}}(K, d)$, which is closed by taking the 0-ball around it).

We show $\mathfrak{B} \neq \mathfrak{A}$.

If t_a is active in $\mathfrak{A}$ then by definition of active, there is $c \in V_t$ (remember that here, $t = t_a$) such that either, for some $A \in \mathbf{Dyn}$ unary, $c \in [\![\Delta A]\!]_{\mathfrak{A}}$, so $[\![\Delta A]\!]_{\mathfrak{A}} \neq [\![\Delta A]\!]_{\mathfrak{A}} \setminus V_t = [\![\Delta A]\!]_{\mathfrak{B}}$; or there is $d \in \mathrm{dom}(\mathfrak{A})$ such that $(c, d) \in [\![\oplus \mathsf{Link}]\!]_{\mathfrak{A}}$, so by definition of $[\![\oplus \mathsf{Link}]\!]_{\mathfrak{B}}$ we get $[\![\Delta \mathsf{Link}^\star]\!]_{\mathfrak{B}} \neq [\![\Delta \mathsf{Link}^\star]\!]_{\mathfrak{A}}$.

Otherwise, we have $(a, b) \in [\![\ominus \mathsf{Link}]\!]_{\mathfrak{A}}$. Since t_a is not active, $a \notin \mathcal{A}_{\bar{t}}(\mathfrak{A})$; and by definition of t, $b \notin \mathcal{A}_{\bar{t}}(\mathfrak{A})$. By assumption $a \notin \mathcal{B}_{\mathfrak{A}}(K, d+1)$ so $a \notin \mathcal{B}_{\mathfrak{A}}(K, d)$, and b is at distance at most 1 from a, so $b \notin \mathcal{B}_{\mathfrak{A}}(K, d)$. We get $[\![\ominus \mathsf{Link}]\!]_{\mathfrak{B}} \neq [\![\ominus \mathsf{Link}]\!]_{\mathfrak{A}}$. □

Using the previous lemma, we obtain that dynamic support is preserved by drawing any protected boundary and clearing a change outside of the boundary.

Lemma 4.3.5. If $\mathfrak{A} : \Theta_n$ is dynamically supported, $d \geqslant 0$, $K \subseteq \mathrm{dom}(\mathfrak{A})$, and there is $A \in \mathbf{Dyn}$, $a \in \mathbf{e} \in [\![\Delta A]\!]_{\mathfrak{A}}$, and $a \notin \mathcal{B}_{\mathfrak{A}}(K, d+1)$, there is a dynamically supported (K, d)-sub $\mathfrak{B}$ of $\mathfrak{A}$ such that $\mathfrak{B} \neq \mathfrak{A}$.

Proof. Immediate application of lemma 4.3.4 since $a \notin \mathcal{B}_{\mathfrak{A}}(K, d+1)$ implies that $a \notin \mathcal{B}_{\mathfrak{A}}(K, d)$. □

We are interested in the size of protected areas, specifically of a ball of radius d around an element. The trees of an n-BFL have a bounded size (height bounded by n, degree bounded by $|\mathbf{Stat}_{\mathrm{fun}}|$) and a bounded number of neighbors (by functionality of Link and $\mathsf{Link}^\star$) so we get an upper bound. Out of convenience, the formulation of the lemma names a witness bounding function: in chapter 7 we will enumerate a number of variables bounded by the size of balls in n-BFLs, and having an explicit bound will simplify notation.

Lemma 4.3.6. There is a function $\ell_{\Theta_n}(d) \in \mathcal{O}(|\mathbf{Stat}_{\mathrm{fun}}|^{2dn^2})$ such that for all $\mathfrak{A} : \Theta_n$ if $a \in \mathrm{dom}(\mathfrak{A})$ and $d \geqslant 0$ then $|\mathcal{B}_{\mathfrak{A}}(a, d)| \leqslant \ell_{\Theta_n}(d)$.

Proof. Trees of $G_{\mathfrak{A}}$ have degree at most $|\mathbf{Stat}_{\mathrm{fun}}| - 1$ and height at most n, so they are of order $\leqslant |\mathbf{Stat}_{\mathrm{fun}}|^n$. Since each element in a tree is linked to at most two other elements (one in Link and one in $\mathsf{Link}^\star$), we can bound the number of trees at distance d from an element by the order of a graph of diameter $2d$ and degree at most $s := 2|\mathbf{Stat}_{\mathrm{fun}}|^n$. By [Miller and Sirán, 2005], that number is bounded above by $\frac{s(s-1)^{2d}-2}{s-2} + 1$ if $n > 0$ and $4d + 1$ if $n = 0$. So $|\mathcal{B}_{\mathfrak{A}}(a, d)| \in \mathcal{O}(|\mathbf{Stat}_{\mathrm{fun}}|^{2dn^2})$ (number of trees $\times$ size of trees). $\square$

Direct application of the previous lemmas shows the importance of preservation:

Theorem 4.3.7. If $\downarrow\phi_1 \vee \ldots \vee \downarrow\phi_m$ is $(\mathcal{V}, d)$-preserved and implies $\mathcal{T}_{ns}$, then for all $d' \geqslant 0$, $\mathcal{P}(\bigvee \downarrow\phi_i, d')$ is finite up to iso.

Proof. It suffices to show that the sup of $|\mathrm{dom}(\mathfrak{A})|$ for $(\mathfrak{A}, K) \in \mathcal{P}(\bigvee \phi_i, d')$ is finite.

Let $(\mathfrak{A}, K) \in \mathcal{P}(\bigvee \phi_i, d')$. For some μ and $1 \leqslant i \leqslant m$, $\mathfrak{A}, \mu \vDash \downarrow\phi_i$. By definition of $\mathcal{P}$,

$$\mathrm{dom}(\mathfrak{A}) = \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d' + 1) = \mathcal{B}_{\mathfrak{A}}(\mathrm{Im}(\mu), d' + 1) \cup \mathcal{B}_{\mathfrak{A}}([\![\Delta]\!]_{\mathfrak{A}}, d' + 1)$$

Since $\mathfrak{A} : \Theta_n$ is supported, by lemma 4.3.5 $[\![\Delta]\!]_{\mathfrak{A}} \subseteq \mathcal{B}_{\mathfrak{A}}(\mu(\mathcal{V}), d + 1)$. Moreover $\mu(\mathcal{V}) \subseteq \mathrm{Im}(\mu)$, so

$$\mathcal{B}_{\mathfrak{A}}([\![\Delta]\!]_{\mathfrak{A}}, d' + 1) \subseteq \mathcal{B}_{\mathfrak{A}}(\mathrm{Im}(\mu), d' + d + 2)$$

therefore

$$\mathrm{dom}(\mathfrak{A}) \subseteq \mathcal{B}_{\mathfrak{A}}(\mathrm{Im}(\mu), d' + d + 2)$$

So by lemma 4.3.6, $|\mathrm{dom}(\mathfrak{A})|$ is bounded by a function of $|\langle \bigvee \phi_i \rangle|$, Θ, d and d'. $\square$

4.4. Building preserved formulas with $\vdash_{\lhd}$

In this section, we show how to construct preserved formula by introduced a set of "typing rules" we call $\vdash_{\lhd}$. First, we present the ruls and give the intuition behind them. Then we prove that $\vdash_{\lhd}$ is correct for preservation. Recall that preservation is a property parameterised by a set of variables $\mathcal{V}$ and some $d \geqslant 0$. Similarly, $\vdash_{\lhd}$ has a context of the form $\mathcal{V}, d$.

4.4.1. The rule system $\vdash_{\lhd}$

Note that in the following rules, the context are parameters of the preservation property. In the sequel, we will show that if $\mathcal{V}, d \vdash_{\lhd} \phi$ then ϕ is $(\mathcal{V}, d)$-preserved.

Definition. If x, y are two variables and $\mathcal{V}$ is a set of variables, $\mathcal{V}^{+y}_{-x} := (\mathcal{V} \cup \{y\}) \setminus \{x\}$. ■

In the rules, $\alpha(x, y)$ is any guard. The preservation property is designed to compose well. The following deduction rules all produce preserved formulas:

$$L \in \mathcal{L}_{\mathbf{Dyn}^\star} \; \frac{}{\langle L \rangle ; 0 \vdash_{\lhd} L}\,\textsc{Dynamic}_{\lhd} \qquad\qquad L \in \mathcal{L}^{=}_{\mathbf{Dyn},\mathbf{Stat}} \; \frac{}{\emptyset ; 0 \vdash_{\lhd} L}\,\textsc{Static}_{\lhd}$$

$$\begin{matrix}\mathcal{V} \subseteq \mathcal{V}' \\ d \leqslant d'\end{matrix} \; \frac{\mathcal{V} ; d \vdash_{\lhd} \phi}{\mathcal{V}' ; d' \vdash_{\lhd} \phi}\,\textsc{Weak}_{\lhd} \qquad\qquad \oplus \in \{\wedge, \vee\} \; \frac{\mathcal{V} ; d \vdash_{\lhd} \phi_1 \qquad \mathcal{V} ; d \vdash_{\lhd} \phi_2}{\mathcal{V} ; d \vdash_{\lhd} \phi \oplus \phi_1}\,\textsc{Bool}_{\lhd}$$

$$\frac{\mathcal{V}\,;d \vdash_{\lhd} \phi}{\langle\phi\rangle\,;d+1 \vdash_{\lhd}{\downarrow}(\phi \wedge \mathit{Support}_{D} \wedge \mathit{Symlink})}\ \textsc{Circum}_{\lhd} \qquad \frac{\{x\};0 \vdash_{\lhd} \phi^{\star}}{\emptyset\,;0 \vdash_{\lhd} \phi^{\circ} \wedge \phi^{\star}}\ \textsc{Invariant}_{\lhd}$$

$$x \notin \mathcal{V}\ \frac{\mathcal{V}\,;d \vdash_{\lhd} \phi}{\mathcal{V}\,;d \vdash_{\lhd} \forall x.\ \phi}\ \forall_{\lhd} \qquad \frac{\mathcal{V}\,;d \vdash_{\lhd} \phi}{\mathcal{V}^{+y}_{-x}\,;d + |\{x\} \cap \mathcal{V}| \vdash_{\lhd} \alpha(x,y) \to \phi}\ \textsc{Guard}_{\lhd}$$

$$\frac{\mathcal{V}\,;d \vdash_{\lhd} \phi}{\mathcal{V}^{+y}_{-x};d+1 \vdash_{\lhd} \exists x.\ \alpha(x,y) \wedge \phi}\ \exists\text{-}\textsc{Guard}_{\lhd}$$

Discussion

The rules $\textsc{Static}_{\lhd}$ and $\textsc{Dynamic}_{\lhd}$ introduce literals. Equality literals, literals on the precondition and static literals do not need any protected area to remain satisfied under sub since a sub only removes changes by altering the *postcondition*. On the other hand, any variable mentioned in a literal from $\mathcal{L}_{\mathbf{Dyn}^{\star}}$ should a priori be protected (the rule can actually be slightly relaxed).

The rule $\textsc{Weak}_{\lhd}$ directly implements the weakening of lemma 4.3.2: if ϕ-satisfaction is closed under taking subs for some protected area, it's always safe to grow that area.

The two rules packed in $\textsc{Bool}_{\lhd}$ show that positive boolean operations are well-behaved with regards to preservation.

At a high level, the rule $\textsc{Circum}_{\lhd}$ is intuitive: if a set is closed for subs, its $\trianglelefteq$-minimal set of $\trianglelefteq$ is as well (remember that the sub relation is finer than $\trianglelefteq$). The formulas $\mathit{Support}_{D}$ and *Symlink* are added in because they are not provable (unlike the rest of $\mathcal{T}_{ns}$). Also, while not necessary for the lemmas and theorems of this chapter, we grow the context in $\textsc{Circum}_{\lhd}$ (adding 1 to d, adding all free variables to $\mathcal{V}$) so that we can prove syntactic properties about $\vdash_{\lhd}$-provable formulas in chapter 7.

More discussion about $\mathit{Support}_{D}$ and *Symlink* is warranted. $\mathit{Support}_{D}$ is not provable, and is indeed not preserved. This is simply because any "appearing" element (so an element in some $[\![\mathsf{Has}^{\star}]\!]_{\mathfrak{A}} \setminus [\![\mathsf{Has}]\!]_{\mathfrak{A}}$) ceases to be supported once it is selected for clearing (when taking a sub). However, put under $\downarrow$, $\mathit{Support}_{D}$ is such that there are no proper subs at all, so the property of preservation becomes vacuously true. As for *Symlink*, that formula actually *is* $(\emptyset, 0)$-preserved, but the deduction rules cannot prove it. In the last chapter we mention some ideas on how to make *Symlink* provable.

The rule $\textsc{Invariant}_{\lhd}$ says that a postcondition property that only requires protecting changes local to a single tree requires no protection at all once that property is extended to the precondition.

Example. Take the formula $A^{\star}(x) \vee B^{\star}(x)$. If $\mathfrak{A}$ is an n-BFL and $a \in \mathrm{dom}(\mathfrak{A})$ such that $\mathfrak{A}, x \mapsto a \vDash A^{\star}(x) \vee B^{\star}(x)$ then any sub of $\mathfrak{A}$ which protects a still satisfies the formula. Now with the formula

$$F = (A(x) \vee B(x)) \wedge (A^{\star}(x) \vee B^{\star}(x))$$

no protection is required: if the changes local to a are cleared, they are cleared all at once. Since $A(a) \vee B(a)$ is satisfied in the precondition, a complete absence of changes means $A^{\star}(a) \vee B^{\star}(a)$ will be satisfied in the postcondition.

We illustrate the effect in figure 4.7. On the top left, a transition satisfies $\forall A^{\star}(x) \vee B^{\star}(x)$. But after clearing a (bottom left), the new transition does not satisfy $\forall A^{\star}(x) \vee B^{\star}(x)$. Compare with the right side: on the top right, a transition satisfies $\forall F$, and the transition that results from clearing a still satisfies $\forall F$. ■

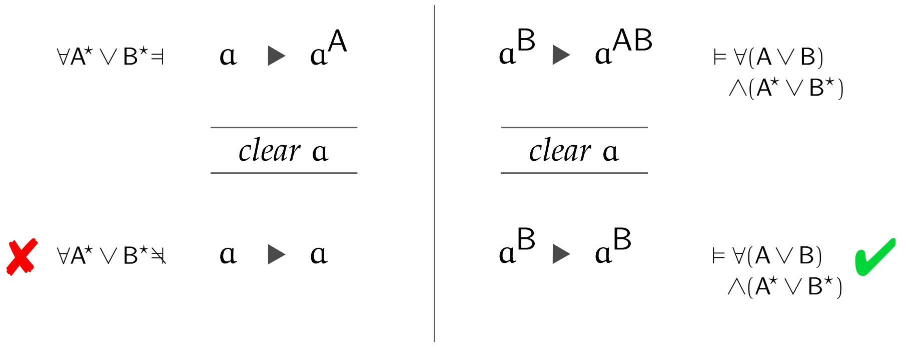


Figure 4.7.: Illustrating the effect of INVARIANT$_{\lhd}$

In tandem with INVARIANT$_{\lhd}$, the rule $\forall_{\lhd}$ can build universal sentences: once a variable x has been removed from the context $\mathcal{V}$ using INVARIANT$_{\lhd}$, x can be universally quantify using $\forall_{\lhd}$ (note the proviso $x \notin \mathcal{V}$).

The rule GUARD$_{\lhd}$ combines two ideas. The first idea is that if we know that the interpretation of two variables x and y are linked, instead of having both x and y in the context, we can keep only y in the context and extend the protected area by increasing d. Again, this means we can universally quantify on x. The second idea is that if $x \notin \mathcal{V}$, we actually only need to protect y and do not need include the interpretation of x in the protection radius. This is thanks to the asymmetry in the definition of subs:

Example. Take the formula $\neg\mathsf{Link}^\star(x, y)$. It is equivalent to a formula of the form $\alpha(x, y) \to \phi$ (where ϕ is $x \neq x$, for instance). Going just by DYNAMIC$_{\lhd}$, the context should be $\{x, y\}, 0$. But by GUARD$_{\lhd}$, the context can be just $\{y\}, 0$; that is, $\neg\mathsf{Link}^\star(x, y)$ is $(\{y\}, 0)$-preserved. This is due to the asymmetry in the definition of subs: link deletions are not cleared when they touch an element of the protected set. However, note that the positive literal $\mathsf{Link}^\star(x, y)$ is *not* $(\{y\}, 0)$-preserved, following the definition of sub which removes any link addition which touches a cleared tree. One may wonder if the same trick used for link deletions could also be used for link additions. We were not able to construct such a symmetric notion of sub; in particular the definition of *active* becomes difficult. In addition we think there is a more fundamental problem, not apparent in the current version of $\vdash_{\lhd}$. As mentioned earlier, *Symlink* is not provable even though it is preserved. With more powerful rules capable of proving formulas like *Symlink*, we foresee that a symmetric version of sub would quickly lead to unbounded changes. ■

Finally, the rule $\exists$-GUARD$_{\lhd}$ acts both as a guard and as a way to add existential quantifiers. The guard part works as in GUARD$_{\lhd}$, except the distance is always increased by 1 because taking a sub can remove an element.

Design note. We think it is possible to build deduction rules with a context of the form $\mathcal{V}, q$, with $q \geqslant 0$, based on a different definition of preservation – instead of taking a ball of radius d around the interpretation of $\mathcal{V}$, one simply asks that it suffices to protect q trees at most in addition those given by $\mathcal{V}$ (which q trees may depend on the interpretation of $\mathcal{V}$). Under this definition, it should be possible to have a rule with unguarded existential quantification. ■

4.4.2. $\vdash_{\lhd}$-provable formulas are preserved

The following lemmas prepare for theorem 4.4.5.

The first one shows that binary dynamic atoms can be treated as unary when the link loops back:

Lemma 4.4.1. If $\mathcal{V}$ is a set of variables, $d \geqslant 0$, and t is a term such that $\langle t \rangle \notin \mathcal{V}$ then the formula $\mathsf{Link}^\star(t,t)$ is not $(\mathcal{V}, d)$-preserved.

Proof. Take a model $\mathfrak{M}$ such that $\mathrm{dom}(\mathfrak{M}) := \{e\}$, an interpretation μ such that $[\![t]\!]_{\mathfrak{M},\mu} = e$, no $[\![\mathsf{Link}]\!]_{\mathfrak{M}} := \emptyset$ and $[\![\mathsf{Link}^\star]\!]_{\mathfrak{M}} := \{(e,e)\}$.

We have $\mathfrak{M}, \mu \models \mathsf{Link}^\star(t,t)$, but a $(\mathcal{V}, d)$-sub of $\mathfrak{M}$ with cleared tree t_e (possible since there is no other variable than $\langle t \rangle$ in $\mathsf{Link}^\star(t,t)$) does not satisfy $\mathsf{Link}^\star(t,t)$. □

This lemma shows that for negative literals on links, it is actually enough to protect only one of the ends of the link. Note that $\vdash_{\triangleleft}$ cannot prove that directly; but the rule Guard$_{\triangleleft}$ exploits that fact.

Lemma 4.4.2. If t, u are terms then $\neg\mathsf{Link}^\star(t,u)$ and $\neg\mathsf{Link}^\star(u,t)$ are $(\{u\}; 0)$-preserved.

Proof. Take $\mathfrak{A}$ and appropriate μ and $\mathfrak{B}$ a $(\{u\}, 0)$-sub of $\mathfrak{A}$ named $\mathfrak{B}$. By definition of a sub, no edge deletion touching the tree of u may be cleared since $[\![u]\!]_{\mathfrak{A},\mu} \in \mathcal{B}_{\mathfrak{A}}([\![\langle u \rangle]\!]_{\mathfrak{A},\mu}, 0)$. □

The next lemma shows that pre formula do not look at $\mathsf{Dyn}^\star$ at all.

Lemma 4.4.3. If ϕ is a pre formula, $\mathfrak{A}, \mu, \nu \models \phi$, $\mathfrak{B} : \Theta$, $\mathrm{Im}(\mu) \cup \mathrm{Im}(\nu) \subseteq \mathrm{dom}(\mathfrak{B})$, $[\![\mathbf{Dyn}]\!]_{\mathfrak{B}} = [\![\mathbf{Dyn}]\!]_{\mathfrak{A}}$ and $[\![\mathbf{Stat}]\!]_{\mathfrak{B}} = [\![\mathbf{Stat}]\!]_{\mathfrak{A}} \upharpoonright \mathrm{dom}(\mathfrak{B})$ then $\mathfrak{B}, \mu, \nu \models \phi$.

Proof. Trivial by induction on the structure of ϕ. □

This lemma shows that $\phi^\star$ is the syntactic counterpart of the mirror operation on pre formulas.

Lemma 4.4.4. If ϕ is a pre formula and $\mathfrak{A}, \mu, \nu, \models \phi$ then $\mathfrak{A}^{-1}, \mu, \nu \models \phi^\star$

Proof. Trivial by induction on the structure of ϕ. □

We now show the correctness of $\vdash_{\triangleleft}$. The proof is by case analysis; no induction on the derivation is required. The interesting cases are Guard$_{\triangleleft}$ and Invariant$_{\triangleleft}$.

For Guard$_{\triangleleft}$, the trick is lemma 4.4.2, which shows that a negative guard $\alpha(x,y)$ only needs to protect one of its variables. Note that this is thanks to the asymmetry between link creation and link deletion in the definition of a sub. As a result, a formula of the form $\alpha(x,y) \to \phi$ may not need to protect both x and y if ϕ does not require it.

The case of Invariant$_{\triangleleft}$ is illustrated to help the reader. The asymmetry in the definition of a sub is again on display: we see no element needs to be protected from clearance but first clearing the only tree that *could* need protection (the tree of the interpretation of x) and then taking the mirror of the structure. Any remaining change involving the tree of interpretation of x is a link addition, and due to the way a sub is defined, any remaining link creation can be cleared *without* explicitly clearing that tree, but just by clearing its neighbors.

Theorem 4.4.5. If $\mathcal{V}; d \vdash_{\triangleleft} \phi$, then ϕ is $(\mathcal{V}, d)$-preserved.

Proof. By case analysis.

Weak$_{\triangleleft}$ Immediate by lemma 4.3.2.

Bool$_{\triangleleft}$ Let $\mathfrak{A} : \Theta_n$, $\mu : \langle \mathcal{V} \rangle \to \mathrm{dom}(\mathfrak{A})$. Let $\mathfrak{B}$ be a $(\mu(\mathcal{V}), d)$-sub of $\mathfrak{A}$. Let $\nu : \langle \phi \rangle \setminus \langle \mathcal{V} \rangle \to \mathrm{dom}(\mathfrak{B})$ be such that $A, \mu, \nu \models \phi_1 \oplus \phi_2$.

If $\oplus = \wedge$, by assumption on ϕ_1 and ϕ_2, we have $\mathfrak{B}, \mu, \nu \models \phi_1 \wedge \phi_2$.

If $\oplus = \vee$, for $i = 1, 2$, if $\mathfrak{A}, \mu, \nu \models \phi_i$ then by assumption on ϕ_i, $\mathfrak{B}, \mu, \nu \models \phi_i$. Either way $\mathfrak{B}, \mu, \nu \models \phi_1 \vee \phi_2$.

Guard$_{\triangleleft}$ Let $d' := d + |\{y\} \cap \mathcal{V}|$. Let $\mu : \mathcal{V}^{+y}_{-x} \to \mathrm{dom}(\mathfrak{A})$, and $\mathfrak{B}$ be a $([\![\mathcal{V}^{+y}_{-x}]\!]_{\mathfrak{A},\mu}, d')$-sub of $\mathfrak{A}$, and $\nu : \langle \alpha(x,y) \to \phi \rangle \setminus \mathcal{V}^{+y}_{-x} \to \mathrm{dom}(\mathfrak{B})$ be such that $\mathfrak{A}, \mu, \nu \models \alpha(x,y) \to \phi$.

Suppose $\mathfrak{B}, \mu, \nu \models \alpha(x,y)$. We show $\mathfrak{A}, \mu, \nu \models \alpha(x,y)$ by showing that $\neg\alpha(x,y)$ is $(\mathcal{V}^{+y}_{-x}, d')$-preserved: if α is equality or Link, this is trivial by definition of a sub. Otherwise, $\alpha(x,y)$ is of the form $\mathsf{Link}^{\star}(t,u)$ with $\{x\} = \langle t \rangle$ or $\{x\} = \langle u \rangle$; so by lemma 4.4.2, $\neg\alpha(x,y)$ is preserved under $\{y\};0$, so by lemma 4.3.2, it is also preserved under $\mathcal{V}^{+y}_{-x}\,; d'$. Now we know $\mathfrak{A}, \mu, \nu \vDash \alpha(x,y)$, so $\mathfrak{A}, \mu, \nu \vDash \phi$. To get $\mathfrak{B}, \mu, \nu \vDash \phi$, we only have to show that $\mathfrak{B}$ is a $((\mu \uplus \nu)(\mathcal{V}), d)$-sub of $\mathfrak{A}$. We consider whether $y \in \mathcal{V}$:

If $y \in \mathcal{V}$ then $d' = d + 1$. By lemma 4.3.1, we only need $\mathcal{B}_{\mathfrak{A}}((\mu \uplus \nu)(\mathcal{V}), d) \subseteq \mathcal{B}_{\mathfrak{A}}([\![\mathcal{V}^{+y}_{-x}]\!]_{\mathfrak{A},\mu}, d+1)$. Since $\mathcal{V} \setminus \mathcal{V}^{+y}_{-x} \subseteq \{x\}$, it suffices to show $\mathcal{B}_{\mathfrak{A}}(\nu(x), d) \subseteq \mathcal{B}_{\mathfrak{A}}([\![\mathcal{V}^{+y}_{-x}]\!]_{\mathfrak{A},\mu,\nu}, d+1)$. Since $\mathfrak{A}, \mu, \nu \models \alpha(x,y)$, $\delta_{\mathfrak{A}}(\nu(x), \mu(y)) \leqslant 1$, which gives us $\mathcal{B}_{\mathfrak{A}}(\nu(x), d) \subseteq \mathcal{B}_{\mathfrak{A}}(\mu(y), d+1)$. Since $\mu(y) \in [\![\mathcal{V}^{+y}_{-x}]\!]_{\mathfrak{A},\mu}$, we are done.

If $y \notin \mathcal{V}$ then $\mathcal{V} \subseteq \mathcal{V}^{+y}_{-x}$ and $d' = d$, so we can immediately apply lemma 4.3.2, to get $\mathfrak{B}, \mu, \nu \models \phi$.

$\forall_{\triangleleft}$ Let $\mathfrak{A} : \Theta_n$ and $\mu : \langle \mathcal{V} \rangle \to \mathrm{dom}(\mathfrak{A})$. Let $\mathfrak{B}$ be a $(\mu(\mathcal{V}), d)$-sub of $\mathfrak{A}$, and $\nu : \langle \phi \rangle \setminus \langle \mathcal{V} \rangle \to \mathrm{dom}(\mathfrak{B})$ be such that $\mathfrak{A}, \mu, \nu \models \forall y.\ \phi$,

For all $a \in \mathrm{dom}(\mathfrak{A})$, $\mathfrak{A}, \mu, \nu, y \mapsto a \models \phi$. Since $\mathrm{dom}(\mathfrak{B}) \subseteq \mathrm{dom}(\mathfrak{A})$, for all $b \in \mathrm{dom}(\mathfrak{B})$, by assumption on ϕ we have $\mathfrak{B}, \mu, \nu, y \mapsto b \models \phi$. So $\mathfrak{B}, \mu, \nu \models \forall y.\ \phi$.

∃-Guard$_{\triangleleft}$ Let $\mu : \mathcal{V}^{+y}_{-x} \to \mathrm{dom}(\mathfrak{A})$, and $\mathfrak{B}$ be a $([\![\mathcal{V}^{+y}_{-x}]\!]_{\mathfrak{A},\mu}, d+1)$-sub of $\mathfrak{A}$, and $\nu : \langle \alpha(x,y) \wedge \phi \rangle \setminus \mathcal{V}^{+y}_{-x} \setminus \{x\} \to \mathrm{dom}(\mathfrak{B})$, $a \in \mathrm{dom}(\mathfrak{A})$ be such that $\mathfrak{A}, \mu, \nu, x \mapsto a \models \alpha(x,y) \wedge \phi$. We show $a \in \mathrm{dom}(\mathfrak{B})$ and $\mathfrak{B}, \mu, \nu, x \mapsto a \models \alpha(x,y) \wedge \phi$. Note how we cannot assume $a \in \mathrm{dom}(\mathfrak{B})$ because of the existential quantifier we are about to add, and that since $x \neq y$ we have $y \in \mathcal{V}^{+y}_{-x}$.

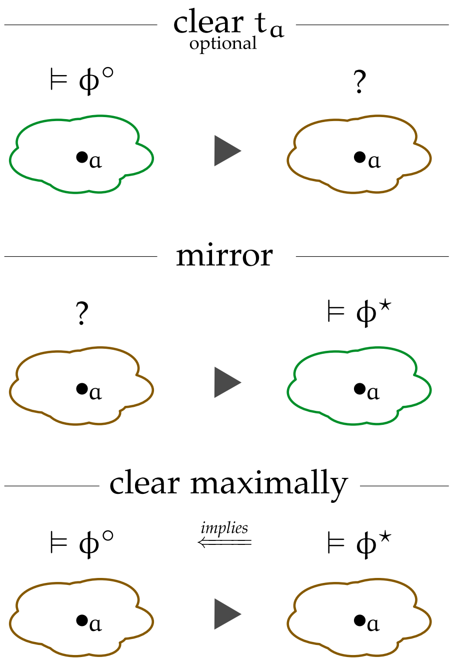
Let $b := \mu(y)$. First, note that since $\delta_{\mathfrak{A}}(a, b) \leqslant 1$ and $y \in \mathcal{V}^{+y}_{-x}$, we have $\mathcal{B}_{\mathfrak{A}}(a, d) \subseteq \mathcal{B}_{\mathfrak{A}}(b, d+1) \subseteq \mathcal{B}_{\mathfrak{A}}([\![\mathcal{V}^{+y}_{-x}]\!]_{\mathfrak{A},\mu}, d+1)$. So, by definition of subs, $a \in \mathrm{dom}(\mathfrak{B})$, and any link between t_a and t_b is preserved in $\mathfrak{B}$, so $\mathfrak{B}, \mu, \nu, x \mapsto a \models \alpha(x,y)$.

For ϕ, as in the case of Guard$_{\triangleleft}$, we need $\mathcal{B}_{\mathfrak{A}}((\mu \uplus x \mapsto a)(\mathcal{V}), d) \subseteq \mathcal{B}_{\mathfrak{A}}([\![\mathcal{V}^{+y}_{-x}]\!]_{\mathfrak{A},\mu}, d+1)$ to apply lemma 4.3.1. Since $y \in \mathcal{V}^{+y}_{-x}$, it would suffice to show $\mathcal{B}_{\mathfrak{A}}(a, d) \subseteq \mathcal{B}_{\mathfrak{A}}(b, d+1)$, which we just established. So $\mathfrak{B}, \mu, \nu, x \mapsto a \models \phi$.

It immediately follows that $\mathfrak{B}, \mu, \nu \models \exists x.\ \alpha(x,y) \wedge \phi$ whenever $\mathfrak{A}, \mu, \nu$ does.

Invariant$_{\triangleleft}$ We know $\{x\}, 0 \vDash_{\triangleleft} \phi^{\star}$. Let $\nu : \langle \phi \rangle \setminus \{x\} \to \mathrm{dom}(\mathfrak{A})$, $\mathfrak{A}, x \mapsto a, \nu \models \phi^{\circ} \wedge \phi^{\star}$ and $\mathfrak{B}$ be a sub of $\mathfrak{A}$ with cleared tree t.

Since ϕ is a pre formula, by lemma 4.4.3, $\mathfrak{B}, x \mapsto a, v \models \phi^\circ$.



Case 1 : If $a \in V_t$ or if t_a is not active in $\mathfrak{A}$, we have that no element of V_{t_a} occurs in any $[\![\Delta A]\!]_{\mathfrak{B}}$, except possibly in edge deletions involving t_a and another tree adjacent to t_a. In the first case it is because $V_{t_a} = V_t$ has just been made inactive by going from $\mathfrak{A}$ to $\mathfrak{B}$, in the second case it is by definition of active. We want $\mathfrak{B}, x \mapsto a, v \models \phi^\star$. The proof will work by flipping Dyn and Dyn* in $\mathfrak{B}$. Any remaining edge deletion that touches t_a will become an edge addition and so will become clearable by taking subs. Then, once all changes have been removed, we can deduce satisfaction of $\phi^\star$ from the satisfaction of ϕ°. Consider $\mathfrak{B}^{-1}$. Since $\mathfrak{B}, x \mapsto a, v \models \phi^\circ$, by lemma 4.4.4. $\mathfrak{B}^{-1}, x \mapsto a, v \models \phi^\star$.
Now, for any $\mathfrak{C} \trianglelefteq \mathfrak{B}^{-1}$ such that $\mathfrak{C}, x \mapsto a, v \models \phi^\star$, we show that if $[\![\Delta]\!]_{\mathfrak{C}} \neq \emptyset$ then there is $\mathfrak{D} \triangleleft \mathfrak{C}$ such that $\mathfrak{D}, x \mapsto a, v \models \phi^\star$ and $\mathfrak{D}$ has strictly fewer changes than $\mathfrak{C}$.

- If there is an active tree t different from t_a, clear it (that is, let $\mathfrak{D}$ be a sub of $\mathfrak{C}$ with cleared tree t and minimal K'). By assumption on ϕ, $\mathfrak{D}, x \mapsto a, v \models \phi^\star$, and clearing an active tree always results in strictly fewer changes by definition.

- Otherwise, note that since t_a is not active in $\mathfrak{B}$, the only changes in $\mathfrak{C}$ that can involve an element $e \in V_{t_a}$ are an edge addition between e and an element $e' \notin V_{t_a}$. So if there is no tree $t \neq t_a$ active in $\mathfrak{C}$, then t_a is not active in $\mathfrak{C}$.

  Therefore no tree is active in $\mathfrak{C}$. So the only possible changes in $\mathfrak{C}$ are removal of some (b, c) edge, with t_a, t_b, t_c pairwise distinct. Let $\mathfrak{D}$ be a sub of $\mathfrak{C}$ with cleared tree t_b and minimal K'. Since t_c is not active and is different from t_a, $\mathfrak{D}$ contains strictly fewer changes than $\mathfrak{C}$. Again by assumption on ϕ, $\mathfrak{D}, x \mapsto a, v \models \phi^\star$.

Using the above statement, by induction on the product ordering on each $[\![\Delta A]\!]_{\mathfrak{B}^{-1}}$, there is $\mathfrak{C} \trianglelefteq \mathfrak{B}^{-1}$ such that $[\![\Delta]\!]_{\mathfrak{C}} = \emptyset$ and $\mathfrak{C}, x \mapsto a, v \models \phi^\star$.
Since $[\![A^\star]\!]_{\mathfrak{C}} = [\![A]\!]_{\mathfrak{C}}$ for every $A \in$ Dyn, $\mathfrak{C}, x \mapsto a, v \models \phi^\circ$.
Since $\mathfrak{C} \trianglelefteq \mathfrak{B}^{-1}$, $[\![A]\!]_{\mathfrak{C}} = [\![A]\!]_{\mathfrak{B}^{-1}}$ for every $A \in$ Dyn. So $\mathfrak{B}^{-1}, x \mapsto a, v \models \phi^\circ$.
Since $[\![A]\!]_{\mathfrak{B}^{-1}} = [\![A^\star]\!]_{\mathfrak{B}}$ for every $A \in$ Dyn, $\mathfrak{B}, x \mapsto a, v \models \phi^\star$.
The side figure illustrates case 1.
Case 2 : Otherwise, $a \notin V_t$ and t_a is active. If t does not share an edge with t_a, $\mathfrak{B}$ is a $(\{a\}, 0)$-sub of $\mathfrak{A}$. If it does, since t_a is active and different from t, it is in $\mathcal{A}_{\bar{t}}(\mathfrak{A})$, so, any edge deletion between t and t_a is preserved, so $\mathfrak{B}$ is still a $(\{a\}, 0)$-sub of $\mathfrak{A}$. Since $\mathfrak{A}, x \mapsto a, v \models \phi^\star$, we have $\mathfrak{B}, x \mapsto a, v \models \phi^\star$.

Static$_\triangleleft$ Trivial since equality, preconditions and static predicates do not change by taking a sub.

Dynamic$_{\triangleleft}$ Unary case: let $\mathsf{A} \in \mathbf{Dyn}$ and take a well-defined $[\![t]\!]_{\mathfrak{A},\mu}$ for some $t, \mathfrak{A}, \mu$. By definition of preservation, its may not be cleared. So in any $([\![\langle t\rangle]\!]_{\mathfrak{A},\mu}, 0)$-sub $\mathfrak{B}$ of $\mathfrak{A}$, $[\![t]\!]_{\mathfrak{A},\mu} \in [\![\mathsf{A}^{\star}]\!]_{\mathfrak{A}}$ iff $[\![t]\!]_{\mathfrak{B},\mu} \in [\![\mathsf{A}^{\star}]\!]_{\mathfrak{B}}$.

Binary case works the same way: take well defined $[\![t]\!]_{\mathfrak{A},\mu}$ and $[\![u]\!]_{\mathfrak{A},\mu}$. No edge with both ends in $\mathcal{B}_{\mathfrak{A}}([\![\langle t\rangle \cup \langle u\rangle]\!], 0)$ may be cleared, so $[\![(t,u)]\!]_{\mathfrak{A},\mu} \in [\![\mathsf{Link}^{\star}]\!]_{\mathfrak{A}}$ iff $[\![(t,u)]\!]_{\mathfrak{B},\mu} \in [\![\mathsf{Link}^{\star}]\!]_{\mathfrak{B}}$.

Circum$_{\triangleleft}$ We show that $\downarrow(\phi \wedge \mathit{Support}_D \wedge \mathit{Symlink})$ is $(\mathcal{V}, d)$-preserved, the rest is by **Weak**$_{\triangleleft}$.

Take some $\mathfrak{A} : \Theta_n$ such that $\mathfrak{A}, \mu \vDash \phi$, let $K := \mu(\mathcal{V})$, and take a $\mathcal{B}_{\mathfrak{A}}(K, d)$-sub $\mathfrak{B}$ of $\mathfrak{A}$. By assumption $\mathfrak{B}, \mu \vDash \phi$. By lemma 4.3.3, $\mathfrak{B} : \Theta_n$, so $\mathfrak{B} \vDash \mathit{Symlink}$. Now suppose $\mathfrak{B}$ is not dynamically supported. In that case, the cleared tree t contains an element $a \in [\![\oplus\mathsf{Has}]\!]_{\mathfrak{A}}$, and that element is outside of $\mathcal{B}_{\mathfrak{A}}(K, d)$, by definition of $\mathfrak{B}$. So by lemma 4.3.4, there is $\mathfrak{B}'$, a (K, d)-sub of $\mathfrak{A}$ which is dynamically supported and such that $\mathfrak{B}' \neq \mathfrak{A}$. Again by 4.3.3 $\mathfrak{B}' \vDash \mathit{Symlink}$, and by assumption, $\mathfrak{B}', \mu \vDash \phi$. By definition of a sub, $\mathfrak{B}' \trianglelefteq \mathfrak{A}$, so $\mathfrak{A}$ was not minimal. Absurd. $\square$

5. A safe program syntax

In this chapter we introduce a syntax for logic programs. Interpreted as formulas, these programs are guaranteed to have their run semantics coincide with the execution of a finite set of rules.

The shape of a program is a sequence of local "reactions" followed by global "invariants". The reactions should be seen as the successor to graph-rewriting rules: local pieces of knowledge that say what mechanistic interactions can happen in the cell. The invariants are universal formulas. They enforce coherence or encoding constraints that should be true at all times during a run.

To show that all programs can be seen as a finite set of rules, we show that the formula induced by a program is both $\vDash_\rho$ and $\vDash_\triangleleft$-provable (up to logical equivalence). In the case of invariants, this requires a small amount of syntactic gymnastics to reach a clausal form where every clause is great.

Notation. In the sequel, if ϕ is a formula:

- $\vDash_\rho \phi$ stands for "there is a context Γ and a formula $\psi \equiv \phi$ such that $\Gamma \vDash_\rho \psi$"
- $\vDash_\triangleleft \phi$ stands for "there is a context $\mathcal{V}, d$ and a formula $\psi \equiv \phi$ such that $\mathcal{V}, d \vDash_\triangleleft \psi$"

■

5.1. Programs P and formulas ϕ_P

In this section we introduce the syntax of programs. Here is an example program :

Definition. A *program* is of the following form:

$$\theta_1 \texttt{ ; ... ; } \theta_k \texttt{ ensure } I_1, \ldots, I_{k'} \texttt{ maintain } I_{k'+1}, \ldots, I_{k''}$$

where the (`ensure` ...) and (`maintain` ...) parts are optional, and the θ_i are formulas generated by the following grammar:

$$\theta ::= L \mid \forall x.\ \alpha(x,y) \to \theta \mid \exists x.\ \alpha(x,y) \wedge \theta \mid \theta \wedge \theta \mid \theta \vee \theta \qquad \text{for all } L \in \mathcal{L}, x, y, \alpha(x,y)$$

and the I_j are pre, quantifier-free formulas of the form

$$I := \alpha_1(x, y_1) \to \ldots \alpha_q(x, y_q) \to \psi \qquad q \geqslant 0$$

such that $\langle\psi\rangle \subseteq \{x, y_1, \ldots, y_q\}$ and for every dynamic literal L in ψ, $\langle L \rangle = \{x\}$.

■

Before we explain how a program corresponds to a formula on transitions, we give an example of a simple program:

Example.

$$\oplus\mathsf{Link}(x.s, y.t) \wedge \mathsf{E}(x) \ ; \ \mathsf{Link}(x.s, y.t) \wedge \oplus\mathsf{Active}(x)$$
$$\texttt{ensure}$$
$$\mathsf{Link}(z.s, w.t) \to \mathsf{E}(z) \to (\mathsf{E}(w) \wedge \mathsf{Open}(z.s))$$

where $x.s, y.t, \dots$ are syntactic sugar for terms like $s(x)$. This program has two rules. The first can add a Link between two s and t sites if the parent of the s site is an Enzyme. The second can make an element Active if its s site is linked to a t site.

In addition, there is an invariant which says that for all elements z, w linked through their respective sites s and t, if z is an Enzyme then w must also be an Enzyme and the site s of z must be Open. ■

We will now define the formulas induced by programs. First, we add some program $\mathtt{P}$ as an implicit paramter:

Implicit parameter $\mathtt{P} \coloneqq \theta_1; \dots; \theta_k$ $\texttt{ensure}$ $I_1, \dots, I_{k'}$ $\texttt{maintain}$ $I_{k'+1}, \dots, I_{k''}$ is a program.

Definition.

$$\mathit{Support}_{0,D} \coloneqq \mathit{Support}_D \wedge \mathit{Support}_0$$

■

As a reminder, $\mathit{Support}_D$ means that a transition is supported and that if an element is in any dynamic property from **Dyn** then it is in Has, and if an element is in any dynamic property from **Dyn*** then it is in Has*. $\mathit{Support}_0$ means that a transition is supported at the level of trees, that each, each tree is at least fully in Has or fully in Has*.

Definition. $\mathtt{P}$ induces a formula, $\phi_\mathtt{P}$:

$$\mathit{Invs} \coloneqq \bigwedge_{1 \leqslant j \leqslant k'} \forall(I_j \wedge I_j^\star) \wedge \bigwedge_{k'+1 \leqslant j \leqslant k''} \forall(I_j \to I_j^\star) \qquad \mathit{Context} \coloneqq \mathcal{T}_n \wedge \mathit{Support}_{0,D} \wedge \mathit{Invs}$$

$$\phi_\mathtt{P} \coloneqq \bigvee_{1 \leqslant i \leqslant k} \downarrow(\theta_i \wedge \mathit{Context})$$

■

In $\mathtt{P}$, each θ_i is a local reaction: every quantification is guarded, so the formula cannot say anything beyond a neighbhood of the interpretation of the variables. The I's are both "state" formulas (in the sense that they either not use **Dyn** or not use **Dyn***) and purely universal. They are separated in two sequences.

Members of the first are interpreted as $(I \wedge I^\star)$, i.e. a constraint that should be true at every step of a run. For instance, suppose an element must at least be phosphorylated to be considered "active". We can write $(\mathsf{Active}(x) \to \mathsf{Phos}(x)) \wedge (\mathsf{Active}(x) \to \mathsf{Phos}(x))^\star$.

Members of the second sequence are interpreted as $(I \to I^\star)$, a constraint that should not vary throughout a run. For instance, suppose that some existing program is not supposed to influence the phosphorylation state of the elements throughout the run. We can write $\mathtt{P} \wedge (\mathsf{Phos} \to \mathsf{Phos}^\star) \wedge (\neg\mathsf{Phos} \to \neg\mathsf{Phos}^\star)$ to make sure that it is the case.

5.2. $\cdot \vDash_\rho \phi_\mathtt{P}$

In this section, we show that program formulas are $\vDash_\rho$-provable. The θ_i part is trivial.

Lemma 5.2.1. For each θ_i, $\models_\rho \theta_i$.

Proof. By induction on the structure: if a literal, use LITERAL$_\rho$, if a disjunction use DISJ$_\rho$, if a conjunction use CONJ$_\rho$. If a guarded universal quantification use $\forall$GUARD$_\rho$, if a guarded existential quantification use $\exists$GUARD$_\rho$. □

The following lemma shows that invariants a $\models_\rho$-provable. Note that the conditions are weaker than those given in the definition of program: the free variables in dynamic literals of ψ do *not* have to all be x.

The difficulty here is that we want unguarded universal quantification, and all we have are universally quantified clauses thanks to the rule CLAUSE$_\rho$. We present the idea with an example. Consider the formula

$$F(x,y) := \mathsf{Link}(x,y) \to (A(x) \wedge K(y))$$

We show $\cdot \models_\rho F \to F^\star$. First, note that

$$\begin{aligned} F(x,y) &\equiv \mathcal{C}_1(x,y) \wedge \mathcal{C}_2(x,y) \\ \mathcal{C}_1(x,y) &:= \neg\mathsf{Link}(x,y) \vee A(x) \\ \mathcal{C}_2(x,y) &:= \neg\mathsf{Link}(x,y) \vee K(y) \end{aligned}$$

And both $G_{\mathcal{C}_1}$ and $G_{\mathcal{C}_2}$ are connected. Now we have

$$\begin{aligned} & F(x,y) \to F^\star(x,y) \\ \equiv\; & (\mathcal{C}_1(x,y) \wedge \mathcal{C}_2(x,y)) \to (\mathcal{C}_1^\star(x,y) \wedge \mathcal{C}_2^\star(x,y)) \\ \equiv\; & \Big(\mathcal{C}_1^\star(x,y) \vee \mathsf{Link}(x,y) \vee \mathsf{Link}(x,y)\Big) \wedge \Big(\mathcal{C}_1^\star(x,y) \vee \mathsf{Link}(x,y) \vee \neg K(y)\Big) && (1) \\ \wedge\; & \Big(\mathcal{C}_1^\star(x,y) \vee \neg A(x) \vee \mathsf{Link}(x,y)\Big) \wedge \Big(\mathcal{C}_1^\star(x,y) \vee \neg A(x) \vee \neg K(y)\Big) && (2) \\ \wedge\; & \Big(\mathcal{C}_2^\star(x,y) \vee \mathsf{Link}(x,y) \vee \mathsf{Link}(x,y)\Big) \wedge \Big(\mathcal{C}_2^\star(x,y) \vee \mathsf{Link}(x,y) \vee \neg K(y)\Big) && (3) \\ \wedge\; & \Big(\mathcal{C}_2^\star(x,y) \vee \neg A(x) \vee \mathsf{Link}(x,y)\Big) \wedge \Big(\mathcal{C}_2^\star(x,y) \vee \neg A(x) \vee \neg K(y)\Big) && (4) \end{aligned}$$

We have avoided making boolean simplification. Each of the clauses above also has a connected graph. For the second clauses lines (3) and (4), $\mathcal{C}_i^\star(x,y)$ contains $K(y)$, so the clause is valid. All the other clauses are great:for those on lines (1) and for the firsts of lines (2),(3) and (4), it is because they contain both $\mathsf{Link}(x,y)$ and $\neg\mathsf{Link}^\star(x,y)$. For the second clauses of lines (2) and (3), it is because they contain both $\neg A(x)$ and $A^\star(x)$. Since they are great, they are provable and the rest is by CONJ$_\rho$ and the rest is because $\forall$ distributes over $\wedge$.

We now give the proof for the general case:

Lemma 5.2.2. If $q \geqslant 0$ and ϕ is a pre, quantifier-free formula of the form

$$\alpha_1(x,y_1) \to \ldots \to \alpha_q(x,y_q) \to \psi$$

such that $\langle\psi\rangle \subseteq \{x, y_1, \ldots, y_q\}$, then $\cdot \models_\rho \forall\phi \to \phi^\star$, $\cdot \models_\rho \forall\phi^\star \to \phi$ and $\cdot \models_\rho \forall\phi \wedge \phi^\star$.

Proof. We first show $\cdot \models_\rho \phi \to \phi^\star$. Let $\mathcal{C}_1', \ldots, \mathcal{C}_k'$ be clauses such that $\psi \equiv \bigwedge_j \mathcal{C}_j'$.

ϕ is equivalent to $\bigwedge_{1 \leqslant j \leqslant k} \mathcal{C}_j$ where $\mathcal{C}_j := \mathcal{C}_j' \vee \bigvee_i \neg\alpha_i(x,y_i)$. Note that due to the proviso on $\langle\psi\rangle$, each $G_{\mathcal{C}_j}$ is connected We can also write each $\mathcal{C}_j$ as $L_{j.1} \vee \ldots \vee Lj.m_j$, with each $L_{j.i}$ a literal and m_j the size of $\mathcal{C}_j$. Now:

$$\begin{aligned}
\phi \to \phi^\star &\equiv \left(\bigwedge_{1\leqslant j\leqslant k} \mathcal{C}_j\right) \to \left(\bigwedge_{1\leqslant j\leqslant k} \mathcal{C}_j^\star\right)\\
&\equiv \bigwedge_{1\leqslant j\leqslant k} \mathcal{C}_j^\star \vee \bigvee_{1\leqslant j\leqslant k} \neg\mathcal{C}_j\\
&\equiv \bigwedge_{1\leqslant j\leqslant k}\ \bigwedge_{1\leqslant i_1\leqslant m_1} \cdots \bigwedge_{1\leqslant i_k\leqslant m_k} \mathcal{C}_j^\star \vee \overline{L_{1.i_1}} \vee \ldots \vee \overline{L_{k.i_k}} \qquad (5.1)
\end{aligned}$$

Take any clause $\mathcal{D}_{j.i_1.\cdots.i_k} := \mathcal{C}_j^\star \vee \overline{L_{1.i_1}} \vee \ldots \vee \overline{L_{k.i_k}}$ in (5.1). Each contains $\mathcal{C}_j^\star$, so $G_{\mathcal{D}_{j.i_1.\cdots.i_k}}$ is connected. Now consider $\overline{L_{j.i_j}}$. By definition $L^\star_{j.i_j} \in \mathcal{C}_j^\star$. If $L_{j.i_j} = L^\star_{j.i_j}$, $\mathcal{D}_{j.i_1.\cdots.i_k} \equiv \top$ and we remove it. Otherwise $\overline{L_{j.i_j}} \in \mathsf{Dyn}(\mathcal{D}_{j.i_1.\cdots.i_k})$ and $\overline{\overline{L^\star_{j.i_j}}} \in \mathsf{Dyn}^\star(\mathcal{D}_{j.i_1.\cdots.i_k})$, so $\mathcal{D}_{j.i_1.\cdots.i_k}$ is great and we keep it. Let $\mathcal{D}_{k_1},\ldots,\mathcal{D}_{k_s}$ be the great clauses we kept.

$$\dfrac{\dfrac{}{\mathcal{D}^\circ_{k_1} \models_\rho \forall\mathcal{D}_{k_1}}\ \textsf{CLAUSE}_\rho \qquad \dfrac{}{\mathcal{D}^\circ_{k_2} \models_\rho \forall\mathcal{D}_{k_2}}\ \textsf{CLAUSE}_\rho}{\mathcal{D}^\circ_{k_1}, \mathcal{D}^\circ_{k_2} \models_\rho (\forall\mathcal{D}_{k_1}) \wedge (\forall\mathcal{D}_{k_2})}\ \textsf{CONJ}_\rho$$

$$\dfrac{\ddots \qquad \dfrac{}{\mathcal{D}^\circ_{k_s} \models_\rho \forall\mathcal{D}_{k_s}}\ \textsf{CLAUSE}_\rho}{\mathcal{D}^\circ_{k_1}, \ldots, \mathcal{D}^\circ_{k_s} \models_\rho (\forall\mathcal{D}_{k_1}) \wedge \ldots \wedge (\forall\mathcal{D}_{k_s})}\ \textsf{CONJ}_\rho$$

By the above proof, we get $\cdot \models_\rho \phi \to \phi^\star$. For $\phi^\star \to \phi$, swap the stars in (5.1): again using $L_{j.i_j}$, each clause is either valid or great and the rest follows.

Finally, each $\mathcal{C}_j^\star$ is great, so (as above) by repeated application of $\textsf{CLAUSE}_\rho$ and $\textsf{CONJ}_\rho$, $\models_\rho \forall\phi^\star$. Since we know $\cdot \models_\rho \forall\phi^\star \to \phi$ and since $\phi \wedge \phi^\star \equiv \phi^\star \wedge (\phi^\star \to \phi)$, we get $\cdot \models_\rho \forall\phi \wedge \phi^\star$ using $\textsf{CONJ}_\rho$. □

Lemma 5.2.3. $\cdot \models_\rho \mathcal{T}_n$

Proof. $\mathcal{T}_n$ contains no dynamic literal and a single bound variable, so when in CNF, each clause is great; repeated application of $\textsf{CLAUSE}_\rho$ and $\textsf{CONJ}_\rho$ yields $\cdot \models_\rho \mathcal{T}_n$, up to equivalence.

For *Symlink* and *Funlink*, we apply lemma 5.2.2 as both are the form $\forall\phi \wedge \phi^\star$ with ϕ as the lemma requires. □

Lemma 5.2.4. $\cdot \models_\rho \phi_P$

Proof. First, $\mathit{Support}_{0,D} \equiv \mathit{Support} \wedge \forall\mathsf{Has}_0 \wedge \mathsf{Has}_0^\star \wedge \mathit{Supp}_D \wedge \mathit{Supp}_D^\star$ and by lemma 5.2.2, $\models_\rho \forall\mathsf{Has}_0 \wedge \mathsf{Has}_0^\star$ and $\models_\rho \forall\mathit{Supp}_D \wedge \mathit{Supp}_D^\star$. Moreover, by lemma 5.2.3, $\cdot \models_\rho \mathcal{T}_n$. By lemma 5.2.1, $\models_\rho \theta_i$ and for each I_j, again by lemma 5.2.2, $\cdot \models_\rho \forall(I_j \wedge I_j^\star)$ and $\cdot \models_\rho \forall(I_j \to I_j^\star)$. Note that (unlike in lemma 5.3.4), context matters since $\models_\rho$ has no weakening and $\textsf{DISJ}_\rho$ is additive. Modulo equivalence to the right of $\models_\rho$, we get:

$$\dfrac{\dfrac{\dfrac{\vdots}{\models_\rho \theta_i} \quad \dfrac{\vdots}{J_1 \models_\rho \mathit{Invs}} \quad \dfrac{\vdots}{J_2 \models_\rho \mathcal{T}_n} \quad \dfrac{\vdots}{J_3 \models_\rho \forall\mathsf{Has}_0 \wedge \mathsf{Has}_0^\star} \quad \dfrac{\vdots}{J_4 \models_\rho \forall\mathit{Supp}_D \wedge \mathit{Supp}_D^\star}}{J_1, J_2, J_3, J_4 \models_\rho \theta_i \wedge \mathit{Invs} \wedge \mathcal{T}_n \wedge \forall\mathsf{Has}_0 \wedge \mathsf{Has}_0^\star \wedge \mathit{Supp}_D \wedge \mathit{Supp}_D^\star}\ \textsf{CONJ}_\rho^{(5)}}{J_1, J_2, J_3, J_4 \models_\rho \downarrow\left(\theta_i \wedge \mathit{Invs} \wedge \mathcal{T}_n \wedge \mathit{Support}_{0,D}\right)}\ \textsf{CIRCUM}_\rho$$

Now let $\phi_i := \downarrow\left(\theta_i \wedge \mathit{Invs} \wedge \mathcal{T}_n \wedge \mathit{Support}_{0,D}\right)$,

$$\dfrac{J_1, J_2, J_3 \vdash_{\rho} \phi_1 \qquad \cdots \qquad J_1, J_2, J_3 \vdash_{\rho} \phi_k}{J_1, J_2, J_3 \vdash_{\rho} \phi_P}\ \mathrm{DISJ}_{\rho}{}^{(k-1)}$$

So by $\textsc{Bool}_{\triangleleft}$, $\cdot \vdash_{\rho} \phi_P$. □

5.3. $\cdot \vdash_{\triangleleft} \phi_P$

Here we show that program formulas are $\vdash_{\triangleleft}$-provable.

Lemma 5.3.1. For each θ_i, $\cdot \vdash_{\triangleleft} \theta_i$.

Proof. By induction on the structure: if a literal, use $\textsc{Dynamic}_{\triangleleft}$ or $\textsc{Static}_{\triangleleft}$, if a conjunction or a disjunction use $\textsc{Bool}_{\triangleleft}$ together with $\textsc{Weak}_{\triangleleft}$ if needed. If a guarded existential quantification use $\exists\text{-}\textsc{Guard}_{\triangleleft}$. Finally, if a guarded universal quantification:

$$\dfrac{\dfrac{\mathcal{V}, d \vdash_{\triangleleft} \theta_i'}{\mathcal{V} \setminus \{x\}, d' \vdash_{\triangleleft} \alpha(x,y) \to \theta_i'}\ \textsc{Guard}_{\triangleleft}}{\mathcal{V} \setminus \{x\}, d' \vdash_{\triangleleft} \forall x.\ \alpha(x,y) \to \theta_i'}\ \forall_{\triangleleft}$$

Where $d' \geqslant d$. □

The following lemma shows that invariants a $\vdash_{\rho}$-provable. Note that the conditions are weaker than those given in the definition of program: the free variables of ψ do not have to be included in $\{x, y_1, \ldots, y_q\}$.

Lemma 5.3.2. If $q \geqslant 0$ and ϕ is a pre, quantifier-free formula of the form

$$\alpha_1(x, y_1) \to \ldots \to \alpha_q(x, y_q) \to \psi$$

such that for every dynamic literal L in ψ, $\langle L \rangle = \{x\}$, then $\cdot \vdash_{\triangleleft} \forall\phi \to \phi^{\star}$ and $\cdot \vdash_{\triangleleft} \forall\phi \wedge \phi^{\star}$.

Proof. We first show $\cdot \vdash_{\triangleleft} \forall\phi \wedge \phi^{\star}$.

We start with $\{x\}, 0 \vdash_{\triangleleft} \psi^{\star}$. Wlog assume ψ is in negation normal form. By induction on the structure of $\psi^{\star}$: If a conjunction or disjunction, apply $\textsc{Weak}_{\triangleleft}$ as needed, then $\textsc{Bool}_{\triangleleft}$. If a literal, apply $\textsc{Dynamic}_{\triangleleft}$ or $\textsc{Static}_{\triangleleft}$. By assumption all dynamic literals have $\{x\}$ as variables.

$$\dfrac{\dfrac{\vdots}{\{x\}, 0 \vdash_{\triangleleft} \psi^{\star}}}{\{x\}, 0 \vdash_{\triangleleft} \alpha_1(x, y_1)^{\star} \to \psi^{\star}}\ \textsc{Guard}_{\triangleleft}$$

$$\vdots\ \textsc{Guard}_{\triangleleft}$$

$$\dfrac{\{x\}, 0 \vdash_{\triangleleft} \phi^{\star}}{\emptyset, 0 \vdash_{\triangleleft} \phi \wedge \phi^{\star}}\ \textsc{Invariant}_{\triangleleft}$$

$$\vdots\ \forall_{\triangleleft}$$

$$\dfrac{}{\emptyset, 0 \vdash_{\triangleleft} \forall\phi \wedge \phi^{\star}}\ \forall_{\triangleleft}$$

By the above proof we are done for $\forall\phi \wedge \phi^\star$.

Now, since ϕ is pre and quantifier-free, repeated applications of $\textsc{Static}_\lhd$ and $\textsc{Bool}_\lhd$ yield $\emptyset, 0 \vDash_\lhd \neg\phi$ (up to equivalence). We get:

$$\dfrac{\dfrac{\dfrac{\vdots}{\emptyset, 0 \vDash_\lhd \neg\phi} \qquad \dfrac{\vdots}{\emptyset, 0 \vDash_\lhd \phi \wedge \phi^\star}}{\emptyset, 0 \vDash_\lhd \neg\phi \vee (\phi \wedge \phi^\star)}\,\textsc{Bool}_\lhd \\ \vdots\, \forall_\lhd}{\emptyset, 0 \vDash_\lhd \forall\neg\phi \vee (\phi \wedge \phi^\star)}\,\forall_\lhd$$

Since $\phi \to \phi^\star \equiv \neg\phi \vee (\phi \wedge \phi^\star)$, we are done. ☐

Lemma 5.3.3. $\cdot \vDash_\lhd \mathcal{T}'_n \wedge \mathit{Funlink}$

Proof. Using $\textsc{Bool}_\lhd$, it suffices to show that each conjunct is $\vDash_\lhd$-provable.

$\mathcal{T}'_n$ is purely universal with no dynamic literal, so there is a quantifier-free formula F such that $\emptyset, 0 \vDash_\lhd F$ (using $\textsc{Static}_\lhd$, $\textsc{Bool}_\lhd$ and $\textsc{Weak}_\lhd$) and $\forall F \equiv \mathcal{T}'_n$. The rest is by $\forall_\lhd$.

For *Funlink*, we can apply lemma 5.3.2 as it has the form

$$\forall(\alpha_1(x,y) \to \alpha_2(x,z) \to y = z) \wedge (\alpha_1(x,y) \to \alpha_2(x,z) \to y = z)^\star$$

and $y = z$ is not a dynamic literal.

☐

Note that *Symlink* is not $\vDash_\lhd$-provable.

Lemma 5.3.4. $\cdot \vDash_\lhd \phi_P$

Proof. First, $\mathit{Support}_{0,D} := \mathit{Support} \wedge \forall \mathsf{Has}_0 \wedge \mathsf{Has}_0^\star \wedge \mathit{Supp}_D \wedge \mathit{Supp}_D^\star$ and by lemma 5.3.2, $\vDash_\lhd$ $\forall \mathsf{Has}_0 \wedge \mathsf{Has}_0^\star$. By lemma 5.3.1, $\vDash_\lhd \theta_i$ and for each I_j, again by lemma 5.3.2, $e \vDash_\lhd \forall(I_j \wedge I_j^\star)$ and $\cdot \vDash_\lhd \forall(I_j \to I_j^\star)$.

Second, $\mathcal{T}_n := \mathcal{T}'_n \wedge \mathit{Funlink} \wedge \mathit{Symlink}$ and $\vDash_\lhd \mathcal{T}'_n \wedge \mathit{Funlink}$ by lemma 5.3.3. Modulo equivalence and ignoring contexts (there are no context restrictions on $\textsc{Bool}_\lhd$ and $\textsc{Circum}_\lhd$), we get:

$$\dfrac{\dfrac{\dfrac{\vdots}{\vDash_\lhd \theta_i} \qquad \dfrac{\vdots}{\vDash_\lhd \mathit{Invs}} \qquad \dfrac{\vdots}{\vDash_\lhd \forall \mathsf{Has}_0 \wedge \mathsf{Has}_0^\star} \qquad \dfrac{\vdots}{\vDash_\lhd \mathcal{T}'_n \wedge \mathit{Funlink}}}{\vDash_\lhd \theta_i \wedge \mathit{Invs} \wedge \forall \mathsf{Has}_0 \wedge \mathsf{Has}_0^\star \wedge \mathcal{T}'_n \wedge \mathit{Funlink}}\,\textsc{Bool}_\lhd^{(4)}}{\vDash_\lhd\downarrow \left(\theta_i \wedge \mathit{Invs} \wedge \mathcal{T}_n \wedge \mathit{Support}_{0,D}\right)}\,\textsc{Circum}_\lhd$$

So by $\textsc{Bool}_\lhd$, $\vDash_\lhd \phi_P$.

☐

5.4. Finite compilation of ϕ_P

Here we show how $\models_{\triangleleft}$ and $\models_{\rho}$-provability of a program formula means that there is a finite set of rules with the same execution semantics as the run semantics of the formula.

This preliminary lemmas shows that at the level of rules, reasoning up to iso is enough: since all operations on rules are defined in terms of morphisms, identities stop being important.

Lemma 5.4.1. If R if a set of rules and $R' = R$ up to iso, then $\mathtt{exec}_{\mathfrak{M}}(R) = \mathtt{exec}_{\mathfrak{M}}(R')$ for any $\mathfrak{M} : \Theta$.

Proof. It suffices to show in one direction, for each $\twoheadrightarrow_R$ step. Suppose $R \ni (\mathfrak{S}_1, K, \mathfrak{S}_2) \overset{h}{\simeq} (\mathfrak{S}_1', K', \mathfrak{S}_2') \in R'$, $f : (\mathfrak{S}_1, K) \to \mathfrak{M}'$ and $\mathfrak{M}'$ is the pushout of $(f, g, \mathfrak{S}_1, \mathfrak{S}_2, \mathfrak{M})$. Let $f' = f \circ h$ and $g' = g \circ h$. By lemma 2.4.3 and 2.4.1, $f : (\mathfrak{S}_1', K') \to \mathfrak{M}$. It is routine to check that $g' : \mathrm{dom}(\mathfrak{S}_2') \setminus \mathrm{dom}(\mathfrak{S}_2) \to \mathcal{U} \setminus \mathrm{dom}(\mathfrak{M})$ is injective and that $\mathfrak{M}'$ is the pushout of $(f', g', \mathfrak{S}_2, \mathfrak{S}_2', \mathfrak{M})$ (just transport the construction of $\mathfrak{M}'$ as the pushout of $(f, g, \mathfrak{S}_1, \mathfrak{S}_1', \mathfrak{M})$ through h). □

Definition. If ϕ is a formula and there is a set of rules R such that $\rho_\emptyset(\phi) = \mathtt{exec}_\emptyset(R)$, ϕ can be *compiled* to R. ■

This is the main theorem of the chapter, and the most important of the thesis. It shows that there is a finite set of rules with execution semantics that correspond exactly to the run semantics of a program. Since the rules are of bounded size and model checking is decidable, the proof is constructive (albeit not very practical).

Theorem 5.4.2. ϕ_P can be compiled to a finite set of rules.

Proof. Let $R := \mathtt{rules}(\mathcal{P}(\phi_P, \|\phi_P\|))$. By lemma 5.4.1 it suffices to show that $R/\simeq$ is finite and that $\rho_\emptyset(\phi_P) = \mathtt{exec}_\emptyset(R)$.

By lemma 5.2.4, $\cdot \models_{\rho} \phi_P$, and $\emptyset \in \mathrm{Init}(\psi)$ for any pre,ψ. Moreover $\phi_P \equiv \phi_P \wedge \mathit{Support}_0$, so by theorem 3.5.1 $\rho_\emptyset(\phi_P) = \mathtt{exec}_\emptyset(R)$.

By lemma 5.3.4, $\cdot \models_{\triangleleft} \phi_P$ so by theorem 4.4.5, ϕ_P is $(\mathcal{V}, d)$-preserved for some $\mathcal{V}, d$. By definition ϕ_P implies $\mathcal{T}_{ns}$ and and is a disjunction of minimised formulas, so by lemma 4.3.7, $\mathcal{P}(\phi_P, \|\phi_P\|)$ is finite up to iso, so by definition of $\mathtt{rules}$, $R/\simeq$ is finite. □

6. (Un)decidability modulo n-BFL

This chapter covers the (un)decidability of satisfability for FO Θ-formulas and the elimination of $\downarrow$ from FO[$\downarrow$].

First, in section 6.1, we show that satisfiability modulo the theory of n-BFL is not decidable. The proof goes through a standard domino reduction.

Second, in section 6.2, we show that satisfiability modulo that theory is decidable in $\exists^*\forall^*$ fragment. Without function symbols, this fragment is called the Bernays-Schönfinkel-Ramsay class and decidability in this class is a classic theorem. All we need to do is show that n-BFLs functions are so restricted that the original proof still through.

The *prenex fragments* are used here. As a quick reminder, $\phi \in \exists^*\forall^*$ means that ϕ (not necessarily in FO) is equivalent to some FO formula ϕ' in prenex normal form such that its quantifier prefix is in the language $\exists^*\forall^*$.

Modulo the theory of n-BFLs, theorem 6.1.6 gives a negative result:

Theorem 6.1.6. FO satisfiability modulo $\mathcal{T}_{ns}$ is undecidable for $n \geqslant 1$.

However, there is a fragment where satisfiability is decidable, given by theorem 6.2.3:

Theorem 6.2.3. For $n \geqslant 0$, satisfiability of formulas in $\exists^*\forall^*$ is decidable modulo $\mathcal{T}_n$.

Which is nice from a querying perspective, since all programs are $\vdash_\triangleleft$-provable and theorem 7.3.8 says:

Theorem 7.3.8. If $\mathcal{V}\,;d \vdash_\triangleleft \phi$, then $\phi \wedge \mathcal{T}_n$ is in both $\exists^*\forall^*$ and $\forall^*\exists^*$.

6.1. Undecidability

We show that FO-satisfiability (finite or not) is undecidable for n-BFLs by a reduction from domino problems. The specific approach is inspired by [Grädel, 1999].

We recall the definitions of domino systems and their tilings below, but first give some intuition: a domino system is given by a set of colored, square dominoes D. Dominoes are available in infinite supply, but there are finitely many colors. They must be tiled over the infinite plane following *horizontal* and *vertical* constraints. Horizontal constraints define which colors can touch horizontally; vertical constraints define which colors can touch vertically. A solution to a domino system is a tiling of the plane with the dominoes of D which respects both horizontal and vertical constraints. The problem of finding tilings for domino systems is undecidable.

Definition. A *domino system* $\mathcal{D}$ is a triple (D, H, V) with D finite and H, V two binary relations on D. $\mathcal{D}$ *tiles* $\mathbb{N}^2$ when there exists a *tiling* $\tau : \mathbb{N}^2 \to D$ such that for all $x, y \in \mathbb{N}$:

1. $(\tau(x,y), \tau(x+1,y)) \in H$
2. $(\tau(x,y), \tau(x,y+1)) \in V$

Moreover, τ is *periodic* with horizontal period $h > 0$ and vertical period $v > 0$ if for all $x, y \in \mathbb{N}$,

$$\tau(x, y) = \tau(x + h, y) = \tau(x, y + v)$$

■

Definition (Conservative reduction class)**.** A recursive fragment of FO X is a *conservative reduction class* if there is a recursive function g from FO to X such that for all FO-formulas ψ, ψ is (finitely) satisfiable iff $g(\psi)$ is (finitely) satisfiable. ■

For more on reduction classes, see [Börger et al., 1997]. Since satisfiability is undecidable in full FO, a conservative reduction class has undecidable satisfiability.

Theorem 6.1.1 (Semi-conservative reduction from the domino problem)**.** Let X be a recursive fragment of FO. X is a conservative reduction class if there exists a recursive function that associates with every domino system $\mathcal{D}$ a formula $\psi_{\mathcal{D}} \in X$ such that

1. If $\mathcal{D}$ admits a periodic tiling of $\mathbb{N}^2$, then $\psi_{\mathcal{D}}$ has a finite model.
2. If $\mathcal{D}$ does not tile $\mathbb{N}^2$, then $\psi_{\mathcal{D}}$ is unsatisfiable.

Proof. This is Corollary 3.1.8 in [Börger et al., 1997], chapter 8, p.91. □

Our class X is the set of FO formulas of the form $\phi \wedge \mathcal{T}_{ns}$, so it is recursive.

Domino theory

Let $\mathcal{D} := (D, H, V)$ be a domino system. We will define a BFL signature that can encode domino systems. Then, we will define a formula which is finitely satisfiable when $\mathcal{D}$ has a periodic tiling, and which is unsatisfiable when $\mathcal{D}$ has *no* tiling.

Definition. The *domino signature of* $\mathcal{D}$ is the BFL signature $\Sigma_{\mathcal{D}} := (\mathbf{Dyn}, \mathbf{Dyn}^\star, \mathbf{Stat})$ where

$$\begin{aligned} \mathbf{Dyn} &:= \mathsf{Has}, \mathsf{Link} \\ \mathbf{Dyn}^\star &:= \mathsf{Has}^\star, \mathsf{Link}^\star \\ \mathbf{Stat} &:= \mathbf{D}, \blacktriangle, \blacktriangleright, \blacktriangledown, \blacktriangleleft, \mathit{parent} \end{aligned}$$

$\mathbf{D}$ is a tuple of unary predicate symbols with underlying set D (the ordering is not important), and $\blacktriangle, \blacktriangleright, \blacktriangledown, \blacktriangleleft$ are unary function symbols. ■

$\mathbf{D}$ names the roots in $\Sigma_{\mathcal{D}}$-structures. Each domino will be represented by a tree. The root will have its color (some element of $\mathbf{D}$) as a property. To represent adjacency, roots will have 4 children (one for each direction). The unary functions $\blacktriangle, \blacktriangleright, \blacktriangledown, \blacktriangleleft$ are for the directions *up, right, down, left*, respectively. We define the set of direction symbols $\Delta := \{\blacktriangle, \blacktriangleright, \blacktriangledown, \blacktriangleleft\}$.

We also will need to express "the inverse direction". Let

$$\blacktriangle^\perp := \blacktriangledown \qquad \blacktriangleright^\perp := \blacktriangleleft$$
$$\blacktriangledown^\perp := \blacktriangle \qquad \blacktriangleleft^\perp := \blacktriangleright$$

We sometimes omit parentheses after directions for readability. For instance, we may write $\blacktriangle x$ instead of $\blacktriangle(x)$.

Grid theory A grid is a graph of trees connected only through their ⧗ or their ▸◂ vertices, and where the ▲ ▶ diagonal leads to the same tree as the ▶ ▲ diagonal.

Grids give us a canvas to set the dominos:

$$\begin{aligned} \mathit{Grid} &:= \forall x.\ \mathit{parent}(x) = x \rightarrow \\ &\qquad \wedge\, \exists u, r, d.\ \mathsf{Link}(\blacktriangle x, \blacktriangledown u) \wedge \mathsf{Link}(\blacktriangleright u, \blacktriangleleft d) && (6.1)\\ &\qquad \wedge\, \mathsf{Link}(\blacktriangleright x, \blacktriangleleft r) \wedge \mathsf{Link}(\blacktriangle r, \blacktriangledown d) && (6.2) \end{aligned}$$

The first line restricts the condition to tree roots (recall that an n-BFL can be seen as a forest of bounded-height trees. Tree vertices are structure elements). For every root a, line (6.1) forces the existence of a link between the up-child of a and the down-child of some other root u (for *up-root*). It also forces the existence of a link between the right-child of a and the left-child of some other root r (for *right-root*). Finally, it forces the existence of a root d (for *diagonal-root*) which completes the pattern. See figure 6.1 for an illustration of the effect of *Grid*.

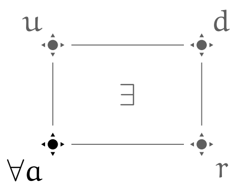


Figure 6.1.: The effect of *Grid*

Definition. For any n-BFL $\mathfrak{A} : \Sigma_{\mathfrak{D}}$ let $\rho_{\mathfrak{A}} \subseteq \mathrm{dom}(\mathfrak{A})$ be the tree roots of $G_{\mathfrak{A}}$. ■

The next lemma provides function in the metalanguage which correspond to going through a diagonal in a domino tiling. Since the lemma shows that the functions traverse any 1-BFL satisfying *Grid* and the way we take the diagonal (right-up or up-right) is irrelevant, it implies that grids are well-formed.

Lemma 6.1.2. For any $\mathfrak{A}$ satisfying $\mathcal{T}_{1S}$ and *Grid*, there are functions $\mathrm{up}_{\mathfrak{A}}, \mathrm{right}_{\mathfrak{A}} : \rho_{\mathfrak{A}} \to \rho_{\mathfrak{A}}$ such that:

- $\mathrm{up}_{\mathfrak{A}} \circ \mathrm{right}_{\mathfrak{A}} = \mathrm{right}_{\mathfrak{A}} \circ \mathrm{up}_{\mathfrak{A}}$
- For all $a \in \rho_{\mathfrak{A}}$ $([\![\blacktriangleright]\!]_{\mathfrak{A}}(a), [\![\blacktriangleleft]\!]_{\mathfrak{A}}(\mathrm{right}_{\mathfrak{A}}(a))) \in [\![\mathsf{Link}]\!]_{\mathfrak{A}}$
- For all $a \in \rho_{\mathfrak{A}}$ $([\![\blacktriangle]\!]_{\mathfrak{A}}(a), [\![\blacktriangledown]\!]_{\mathfrak{A}}(\mathrm{up}_{\mathfrak{A}}(a))) \in [\![\mathsf{Link}]\!]_{\mathfrak{A}}$

Proof. For $a \in \rho_{\mathfrak{A}}$, by $\mathit{Grid}(6.1), \mathit{Grid}(6.2)$ there is $u_a \in \mathrm{dom}(\mathfrak{A})$ such that $\mathfrak{A} \models \mathsf{Link}(\blacktriangle a, \blacktriangledown u_a)$. Since $[\![\mathsf{Link}]\!]_{\mathfrak{A}}$ is functional, u_a is uniquely defined.

Similarly, there are unique r_a and d_a such that $\mathfrak{A} \models \mathsf{Link}(\blacktriangleright u_a, \blacktriangleleft d_a)$, $\mathfrak{A} \models \mathsf{Link}(\blacktriangleright a, \blacktriangleleft r_a)$, and $\mathfrak{A} \models \mathsf{Link}(\blacktriangle r_a, \blacktriangledown d_a)$. We define

$$\begin{aligned} \mathrm{right}_{\mathfrak{A}}(a) &:= r_a \\ \mathrm{up}_{\mathfrak{A}}(a) &:= u_a \end{aligned}$$

Now for the commutativity, we have by definition $\mathrm{right}_{\mathfrak{A}}(u_a) = d_a$ and $\mathrm{up}_{\mathfrak{A}}(r_a) = d_a$, so $\mathrm{right}_{\mathfrak{A}}(\mathrm{up}_{\mathfrak{A}}(a)) = \mathrm{up}_{\mathfrak{A}}(\mathrm{right}_{\mathfrak{A}}(a))$.

□

The next lemma uses $\mathrm{right}_{\mathfrak{A}}$ and $\mathrm{up}_{\mathfrak{A}}$ to define a new function which directly connects the plane and 1-BFLs that satisfy *Grid*.

Lemma 6.1.3. If $\mathfrak{A} \models \mathcal{T}_{1S} \wedge \mathit{Grid}$, there is $\mathfrak{h} : \mathbb{N}^2 \to \rho_{\mathfrak{A}}$ such that for $i, j \in \mathbb{N}$,

$$\mathfrak{h}(i+1, j) = \mathrm{right}_{\mathfrak{A}}(\mathfrak{h}(i,j))$$
$$\mathfrak{h}(i, j+1) = \mathrm{up}_{\mathfrak{A}}(\mathfrak{h}(i,j))$$

Proof. Take any $a \in \rho_{\mathfrak{A}}$. Define $\mathfrak{h}$ for $i, j \geqslant 0$:

$$\mathfrak{h}(i,j) := \mathrm{right}^{i}_{\mathfrak{A}}(\mathrm{up}^{j}_{\mathfrak{A}}(a))$$

Were the ^{i} and ^{j} exponents denote repeated function application. We have:

$$\mathfrak{h}(i+1, j) = \mathrm{right}^{i+1}_{\mathfrak{A}}(\mathrm{up}^{j}_{\mathfrak{A}}(a)) = \mathrm{right}_{\mathfrak{A}}(\mathfrak{h}(i,j))$$
$$\mathfrak{h}(i, j+1) = \mathrm{right}^{i}_{\mathfrak{A}}(\mathrm{up}^{j+1}_{\mathfrak{A}}(a)) = \mathrm{up}_{\mathfrak{A}}(\mathrm{right}^{i}_{\mathfrak{A}}(\mathrm{up}^{j}_{\mathfrak{A}}(\mathfrak{h}(a)))) = \mathrm{up}_{\mathfrak{A}}(\mathfrak{h}(i,j))$$

With the central equality on line 2 making use of lemma 6.1.2. □

Domino theory Once a grid is well-formed, it should contain dominos that tile the plane according to the *horizontal constraints* H and the *vertical constraints* V: a coherent grid w.r.t. V and H ensures that nodes connected through ⧗ respect V and those connected through ▸◂ respect H.

The theory associated to $\mathcal{D}$ is *Domino*. Note that this is the first time formulas are influenced by $\mathcal{D}$ (however, the signature $\Sigma_{\mathcal{D}}$ already depended on $\mathcal{D}$).

$$\mathit{Domino} := \forall x.\ \mathit{parent}(x) = x \to$$
$$\left(\mathsf{Horizontal}(x) \wedge \mathsf{Vertical}(x) \wedge \bigvee_{N \in D} N(x) \wedge \bigwedge_{N' \neq N} \neg N'(x)\right)$$
$$\mathsf{Horizontal}(x) := \bigwedge_{N \in D} N(x) \to \bigvee_{(N,N') \in H} \exists y.\ \mathsf{Link}(\blacktriangleright x, \blacktriangleleft y) \wedge N'(y)$$
$$\mathsf{Vertical}(x) := \bigwedge_{N \in D} N(x) \to \bigvee_{(N,N') \in V} \exists y.\ \mathsf{Link}(\blacktriangle x, \blacktriangledown y) \wedge N'(y)$$

As before for well-formed grids, we define a function in the metalanguage which maps a well-tiled plane with a well-tiled grid:

Lemma 6.1.4. If $\mathfrak{A} : \Sigma_{\mathcal{D}}$ satisfies $\mathcal{T}_{1S}$, *Grid* and *Domino*, there is a function $\tau_{\mathfrak{A}} : \rho_{\mathfrak{A}} \to D$ such that for all $a \in \rho_{\mathfrak{A}}$:

1. $(\tau_{\mathfrak{A}}(a), \tau_{\mathfrak{A}}(\mathrm{right}_{\mathfrak{A}}(a))) \in H$
2. $(\tau_{\mathfrak{A}}(a), \tau_{\mathfrak{A}}(\mathrm{up}_{\mathfrak{A}}(a))) \in V$

Proof. Take any $a \in \rho_{\mathfrak{A}}$. By *Domino*, there is a single $N \in D$ such that $a \in [\![N]\!]_{\mathfrak{A}}$. Define $\tau_{\mathfrak{A}}(a) := N$.

For any $a \in \rho_{\mathfrak{A}}$,with $\phi(x) := \mathsf{Link}(\blacktriangleright a, \blacktriangleleft x)$, by lemma 6.1.2 we have $\mathfrak{A} \models \phi(\mathrm{right}_{\mathfrak{A}}(a))$ and by functionality of Link that is only the case for $\mathrm{right}_{\mathfrak{A}}(a)$. So by $\mathsf{Horizontal}_H$, $\mathfrak{A} \models N'(\mathrm{right}_{\mathfrak{A}}(a))$ for some N' such that $(N, N') \in H$. By definition of $\tau_{\mathfrak{A}}$, $\tau_{\mathfrak{A}}(\mathrm{right}_{\mathfrak{A}}(a)) = N'$, so we are done.

The case for V is similar.

□

Domino model Suppose $\mathcal{D}$ admits a periodic tiling τ with horizontal period h and vertical period v. Let $\mathfrak{T}$ be the $\Sigma_{\mathcal{D}}$-structure with domain of size $5hv$.

$$\mathsf{dom}(\mathfrak{T}) := \{[i,j],\ [i,j].\delta \mid 1 \leqslant i \leqslant h, 1 \leqslant j \leqslant v, \delta \in \Delta\}$$
$$\mathsf{Has}(\mathfrak{T}) := \mathsf{T}$$

Note that "$[i,j]$", "$[i,j].\blacktriangleleft$", and so on are just suggestively-named symbols. For $1 \leqslant i \leqslant h$, $1 \leqslant j \leqslant v$, and $\delta, \delta' \in \Delta$ we specify the function $[\![\delta]\!]_{\mathfrak{T}}$ to be:

$$[\![\delta]\!]_{\mathfrak{T}}([i,j]) := [i,j].\delta$$
$$[\![\delta]\!]_{\mathfrak{T}}([i.j].\delta') := [i,j].\delta'$$

and $[\![\mathsf{Link}]\!]_{\mathfrak{T}}$ to be the smallest symmetric relation such that:

$$([i,j].\blacktriangleright, [i+1 \bmod h, j].\blacktriangleleft) \in [\![\mathsf{Link}]\!]_{\mathfrak{T}}$$
$$([i,j].\blacktriangle, [i, j+1 \bmod v].\blacktriangledown) \in [\![\mathsf{Link}]\!]_{\mathfrak{T}}$$

and let $[\![\mathit{parent}]\!]_{\mathfrak{T}}$ be the parent relation of $G_{\mathfrak{T}}$ on non-roots and the identity on roots; this is well defined since $G_{\mathfrak{T}}$ is trivially a tree.

This gives us a discrete h by v torus. Next, we define the predicates $[\![\mathsf{N}]\!]_{\mathfrak{T}}$ for $\mathsf{N} \in \mathbf{D}$ according to τ. Remembering that by definition, there is a $\mathsf{N} \in \mathbf{D}$ for every $\mathbf{N} \in \mathbf{D}$ and vice versa, we define

$$[i,j] \in [\![\mathsf{N}]\!]_{\mathfrak{T}} \text{ iff } \tau(i \bmod h, j \bmod v) = \mathsf{N}$$

Lemma 6.1.5. $\mathfrak{T} \models \mathcal{T}_{1S} \wedge \mathit{Grid} \wedge \mathit{Domino}$

Proof.

$\mathfrak{T} \models \mathcal{T}_{1S}$ It suffices to show that $\mathfrak{T}$ is a supported 1-BFL.

$\mathfrak{A}$ is a supported n-BFL with symmetric and functional link relations.

Trivially $\mathfrak{T}$ is a 1-BFL with roots $[i,j]$ and leafs $[i,j].\delta$ with all elements in $[\![\mathsf{Has}]\!]_{\mathfrak{T}}$. $[\![\mathsf{Link}]\!]_{\mathfrak{T}}$ is symmetric by definition. Take some a of the form $[i,j].\delta$. If $\delta \in \{\blacktriangleright, \blacktriangle\}$, symmetrisation had no impact and there is a single b such that $(a,b) \in [\![\mathsf{Link}]\!]_{\mathfrak{T}}$. If $\delta = \blacktriangleleft$, there is no $1 \leqslant i' \neq i \leqslant h$ such that $i+1 \bmod h = i'+1 \bmod h$ since i itself ranges from 1 to h, so a has a single image in $[\![\mathsf{Link}]\!]_{\mathfrak{T}}$. The case is similar when $\delta = \blacktriangledown$.

$\mathfrak{T} \models$ ***Grid*** Any root is of the form $[i,j]$ so by definition of $[\![\mathsf{Link}]\!]_{\mathfrak{T}}$ and the four $[\![\delta]\!]_{\mathfrak{T}}$, the desired witnesses u, r, d are respectively $[i, j+1 \bmod v]$, $[i+1 \bmod h, j]$, and $[i+1 \bmod h, j+1 \bmod v]$.

$\mathfrak{T} \models$ ***Domino*** First, roots belong to a single $\mathsf{N}_{\mathfrak{T}}$ predicate because τ is h,v-periodic, so the last part of *Domino* is satisfied.

For Horizontal, only an element of the form $[i,j]$ may have a label N from D. The desired witness y is provided by $[i+1 \bmod h, j]$: by definition of $[\![\blacktriangleright]\!]_{\mathfrak{T}}$, $[\![\blacktriangleleft]\!]_{\mathfrak{T}}$ and $[\![\mathsf{Link}]\!]_{\mathfrak{T}}$ we have $([\![\blacktriangleright]\!]_{\mathfrak{T}}[i,j], [\![\blacktriangleleft]\!]_{\mathfrak{T}}[i+1 \bmod h, j]) \in [\![\mathsf{Link}]\!]_{\mathfrak{T}}$ and since τ tiles $\mathcal{D}$ and is h,v-periodic, $[i+1 \bmod h, j] \in [\![\mathsf{N}']\!]_{\mathfrak{T}}$ for some N' such that $(\mathsf{N}, \mathsf{N}') \in \mathsf{H}$.

The case for Vertical is similar.

□

Theorem 6.1.6. FO satisfiability modulo $\mathcal{T}_{ns}$ is undecidable for $n \geqslant 1$.

Proof. Let $\mathcal{D} = (D, H, V)$ be a domino system. Let $\psi := \mathcal{T}_{1S} \wedge \mathit{Grid} \wedge \mathit{Domino}$ be a formula under signature $\Sigma_{\mathcal{D}}$. To make use of theorem 6.1.1, we first note that the finite structure $\mathfrak{A}(\tau)$ satisfies ψ by lemma 6.1.5.

Then suppose $\mathfrak{A} \models \psi$ for some $\Sigma_{\mathcal{D}}$-structure $\mathfrak{A}$. By lemmas 6.1.3 and 6.1.4 we can define $\tau : \mathbb{N}^2 \to D$ by $\tau(i,j) := \tau_{\mathfrak{A}}(\mathfrak{h}(i,j))$ and obtain:

$$(\tau(i,j), \tau(i+1,j)) = (\tau_{\mathfrak{A}}(\mathfrak{h}(i,j)), \tau_{\mathfrak{A}}(\mathfrak{h}(i+1,j))) = (\tau_{\mathfrak{A}}(\mathfrak{h}(i,j)), \tau_{\mathfrak{A}}(\mathrm{right}_{\mathfrak{A}}(\mathfrak{h}(i,j)))) \in H$$
$$(\tau(i,j), \tau(i,j+1)) = (\tau_{\mathfrak{A}}(\mathfrak{h}(i,j)), \tau_{\mathfrak{A}}(\mathfrak{h}(i,j+1))) = (\tau_{\mathfrak{A}}(\mathfrak{h}(i,j)), \tau_{\mathfrak{A}}(\mathrm{right}_{\mathfrak{A}}(\mathfrak{h}(i,j)))) \in V$$

So $\mathcal{D}$ tiles $\mathbb{N}^2$ with τ. □

6.2. Decidable class

In this section we show that modulo $\mathcal{T}_{ns}$, formulas in $\exists^*\forall^*$ have decidable satisfiability.

Definition. Given a signature Σ with Herbrand universe $\mathcal{W}$ and $\equiv$ an equivalence relation on $\mathcal{W}$, a $\equiv$-*Herbrand* structure $\mathfrak{A}$ is a Σ-structure with universe $\mathcal{W}/\equiv$ such that for all constants c of Σ, $[\![c]\!]_{\mathfrak{A}} = [c]_{\equiv}$, and for all function symbols f of Σ with arity k and all ground terms $t_1, ..., t_k$, $[\![f]\!]_{\mathfrak{A}}([t_1]_{\equiv}, ..., [t_k]_{\equiv}) = [f(t_1, ..., t_k)]_{\equiv}$. A $\equiv$-Herbrand structure that satisfies a formula is a $\equiv$-*Herbrand* model for it. ■

Lemma 6.2.1 (Folklore)**.** A sentence with equality in Skolem normal form is satisfiable iff it has a $\equiv$-Herbrand model.

Proof. A sentence satisfied by a $\equiv$-Herbrand structure is satisfiable. Now assume a sentence in Skolem normal form ϕ satisfied by a structure $\mathfrak{A}$. For any ground Σ-term t, $[\![t]\!]_{\mathfrak{A}}$ is its image in $\mathfrak{A}$.

We proceed by induction on the number n of universal quantifiers in ϕ.

Take the equivalence relation on $\mathcal{W}$ defined by $t \equiv t'$ whenever $[\![t]\!]_{\mathfrak{A}} = [\![t']\!]_{\mathfrak{A}}$ and let $\mathfrak{A}'$ be the $\equiv$-Herbrand structure where for every relational symbol R of Σ with arity k, and any k-tuple of ground terms $(t_1, ..., t_k)$, $([t_1]_{\equiv}, ..., [t_k]_{\equiv}) \in [\![R]\!]_{\mathfrak{A}'}$ iff $([\![t_1]\!]_{\mathfrak{A}}, ..., [\![t_k]\!]_{\mathfrak{A}}) \in [\![R]\!]_{\mathfrak{A}}$. This is well-defined since $\equiv$ on $\mathcal{W}$ respects equality on $\mathfrak{A}$'s universe.

If $n = 0$, note that the truth value of all atoms (equality and predicates) is the same in $\mathfrak{A}'$ and $\mathfrak{A}$ by construction. If $\mathfrak{A} \models \forall x.\ \psi(x)$ with the induction hypothesis assumed on ψ, we have by substitution $\mathfrak{A} \models \psi(t)$ for all ground terms t, by induction $\mathfrak{A}' \models \psi(t)$ for all ground terms t, and thus $\mathfrak{A}' \models \forall x.\ \psi(x)$. □

Lemma 6.2.2. If $\Sigma = (\Sigma', \mathbf{c})$ where Σ' is a BFL signature and $\mathbf{c}$ are constant symbols, if $\mathfrak{A} : \Sigma$ is a $\equiv$-Herbrand model of $\mathcal{T}_n$ then $\mathfrak{A}$ is finite.

Proof. We show: if $\mathfrak{A} \models Tn$ then $\equiv$ is of finite index.

For any constant c and any ground term t of Σ, if t contains c then $[c]_{\equiv}$ and $[t]_{\equiv}$ are in the same tree of $G_{\mathfrak{A}}$. This is easily shown by structural induction on t. There are finitely many c in Σ, so there are finitely many trees in $G_{\mathfrak{A}}$.

Additionally, by lemma 4.1.1, each tree of $G_{\mathfrak{A}}$ is finite. So $G_{\mathfrak{A}}$ is finite. $\mathfrak{A}$ has the same domain, so it is as well. □

We show decidability under a triple restriction : syntactic fragment, signature class, and theory.

Theorem 6.2.3. For $n \geqslant 0$, satisfiability of formulas in $\exists^*\forall^*$ is decidable modulo $\mathcal{T}_n$.

Proof. First, note that $\mathcal{T}_n \in \exists^*\forall^*$. For $\phi \in \exists^*\forall$, $\psi \coloneqq \phi \wedge \mathcal{T}_n$ can equisatisfiably be turned into a Skolem normal form sentence ψ' (replacing variables with constants). By lemmas 6.2.1 and 6.2.2, ψ' is satisfiable iff it is finitely satisfiable. □

7. Change minimality in FO

We then move on to definability questions. In section 7.1, we introduce *unified circumscription*. Generally speaking, circumscription is a 2nd-order schema parameterised by an order o on a class of structures. It takes as input a formula and produces a 2nd-order formula which has, as models, the o-minimal models of ϕ. There are various forms of circumscription; here we introduce a reasonably general variant which suits our purposes.

Then in section 7.2, we show how the operator $\downarrow$ can be expressed by instantiating unified circumscription on the $\trianglelefteq$ order. This provides a syntactic, 2nd-order definition of $\downarrow$. The following section (7.3.4) is about rewriting this formula to eliminate its 2nd-order quantifiers under some conditions.

Finally, we show that the theory of n-BFLs (and in particular the functionality of Link and $\mathsf{Link}^\star$) brings even more simplification: $\vdash_p$-provable minimised formula are definable in the $\exists^*\forall^*$ and $\forall^*\exists^*$ fragments of first-order logic.

7.1. Unified circumscription

In this section, we define unified circumscription syntactically and prove that for an instanciation of unified circumscription on a given order, the models of the circumscribed formula are indeed the minimal models of the original formula along that order.

We now work in second-order logic.

Generally speaking, the purpose of circumscription is to denote through second-order means the minimal models of some FO formula, along a given order. For instance, McCarthy's domain circumscription [McCarthy, 1980] minimises the size of the domain but maintains every predicate and function interpretation on the remaining elements. Also originated by McCarthy, predicate circumscription [McCarthy, 1986] minimises the extent of some *minimised predicates* – but leaves the other predicates and the domain untouched. There are many forms of circumscription; the one we introduce here combines general domain circumscription [Doherty et al., 1998] and parallel predicate circumscription (predicate circumscription extended to the product of subset ordering for multiple predicate interpretations).

With unified circumscription, a signature Σ is partitioned into tuples of predicates and functions. Some predicates are *minimised*. The domain is *minimised* as well. Some predicates are *fixed*: they may not change at all. Other predicates, as well as some functions, are *restricted*: they may not change on the minimised domain but may reduce their domain of definition if the domain is reduced. Finally, some predicates and functions are *varying*: they may change in any way. As we will see, once a partition of Σ has been established, this gives rise to an order $\preceq$ on Σ-structures. Then, given a Σ-formula ϕ, the *circumscription of* ϕ is simply the set of $\preceq$-minimal models of ϕ.

In this section, Σ is assumed to be a signature of the form $(\mathbf{P}_{\text{min}}, \mathbf{P}_{\text{restr}}, \mathbf{f}_{\text{restr}}, \mathbf{P}_{\text{var}}, \mathbf{f}_{\text{var}}, \mathbf{P}_{\text{fix}})$ where the $\mathbf{P}_\bullet$ are predicate tuples and the $\mathbf{f}_\bullet$ are function tuples. We will define a partial order on Σ-models where :

- domains and predicates in $\mathbf{P}_{\text{min}}$ are minimised

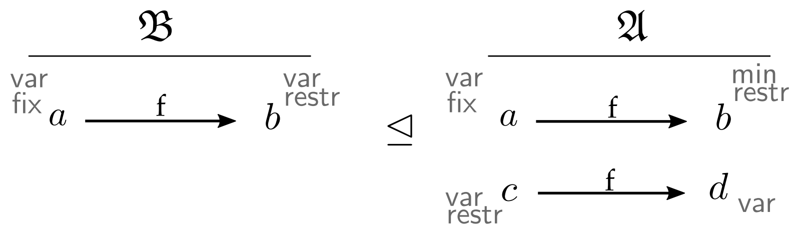


Figure 7.1.: Unified circumscription example.

- predicates in $\mathbf{P}_{\text{restr}}$ and functions in $\mathbf{f}_{\text{restr}}$ are restricted
- predicates in $\mathbf{P}_{\text{var}}$ and functions in $\mathbf{f}_{\text{var}}$ can vary
- predicates in $\mathbf{P}_{\text{fix}}$ are fixed

Definition. If $\mathfrak{A}, \mathfrak{B} : \Sigma$, $\mathfrak{A} \preceq \mathfrak{B}$ whenever :

(a) $\mathsf{dom}(\mathfrak{A}) \subseteq \mathsf{dom}(\mathfrak{B})$

(b) $\llbracket \mathbf{P}_{\text{min}} \rrbracket_{\mathfrak{A}} \subseteq \llbracket \mathbf{P}_{\text{min}} \rrbracket_{\mathfrak{B}}$

(c) $\llbracket \mathbf{P}_{\text{restr}} \rrbracket_{\mathfrak{A}} = \llbracket \mathbf{P}_{\text{restr}} \rrbracket_{\mathfrak{B}} \restriction \mathsf{dom}(\mathfrak{A})$

(d) $\llbracket \mathbf{f}_{\text{restr}} \rrbracket_{\mathfrak{A}} = \llbracket \mathbf{f}_{\text{restr}} \rrbracket_{\mathfrak{B}} \restriction \mathsf{dom}(\mathfrak{A})$

(e) $\llbracket \mathbf{P}_{\text{fix}} \rrbracket_{\mathfrak{A}} = \llbracket \mathbf{P}_{\text{fix}} \rrbracket_{\mathfrak{B}}$

■

There is no condition of $\mathbf{P}_{\text{var}}$ and $\mathbf{f}_{\text{var}}$.

To give some intuition, we try to make the domain and predicates in $\mathbf{P}_{\text{min}}$ as small as possible *given that* $\mathbf{P}_{\text{fix}}$ must not change at all, and that $\mathbf{P}_{\text{restr}}$ and $\mathbf{f}_{\text{restr}}$ must not change on the remaining domain elements.

Example. Let the tuples that partition Σ each have a single predicate: $\mathbf{P}_{\text{min}} = (\mathsf{min}), \mathbf{P}_{\text{restr}} = (\mathsf{restr}), \mathbf{P}_{\text{var}} = (\mathsf{var}), \mathbf{P}_{\text{fix}} = (\mathsf{fix}), \mathbf{f}_{\text{restr}} = (\mathsf{f})$, and consider the following formula and models:

$$\begin{aligned}
\phi &:= \forall x.\ \mathsf{min}(x) \lor \mathsf{var}(x) \\
\mathfrak{A} &:= (\{a, b, c, d\}, \mathsf{min} = \{a\}, \mathsf{fix} = \{a\}, \mathsf{var} = \{b, c, d\}, \mathsf{restr} = \{b, c\} \\
&\qquad \mathsf{f} = a \mapsto b, b \mapsto b, c \mapsto d, d \mapsto d) \\
\mathfrak{B} &:= (\{a, b\}, \mathsf{min} = \emptyset, \mathsf{fix} = \{a\}, \mathsf{var} = \{a, b\}, \mathsf{restr} = \{b\}, \mathsf{f} = a \mapsto b, b \mapsto b)
\end{aligned}$$

As illustrated in figure 7.1, we have $\mathfrak{B} \prec \mathfrak{A}$, and as a matter of fact $\mathfrak{B}$ is $\preceq$-minimal among models of ϕ. $a \in \llbracket \mathsf{fix} \rrbracket_{\mathfrak{A}}$, so circumscription may not lose a, and since $\llbracket \mathsf{f} \rrbracket_{\mathfrak{A}}(a) = b$ and f is restricted, b may not disappear either. On the other hand, domain minimisation drops both c and d: $c, d \notin \mathsf{dom}(\mathfrak{B})$. We have $b \in \llbracket \mathsf{min} \rrbracket_{\mathfrak{A}}$, but by ϕ $b \notin \llbracket \mathsf{min} \rrbracket_{\mathfrak{B}}$ is allowed as long as $b \in \llbracket \mathsf{var} \rrbracket_{\mathfrak{B}}$. Since min is minimised, this is what happens. Finally, unlike fix, restr does not prevent its members from being removed ($c \notin \mathsf{dom}(\mathfrak{B})$), but must remain unchanged on the circumscribed domain (so $b \in \llbracket \mathsf{restr} \rrbracket_{\mathfrak{B}}$ and $a \notin \llbracket \mathsf{restr} \rrbracket_{\mathfrak{B}}$). ■

Definition (Similarity)**.** Two signatures Σ, Σ' are *similar* when they have the same length and symbols in the same position have the same nature (relational or functional) and the same arity. ■

Definition (Substitution)**.** For any two symbols X, Y, $\{X \mapsto Y\}$ is the syntactic map from X to Y, and for any formula ϕ, $\phi\{X \mapsto Y\}$ is ϕ with every instance of X replaced by a Y. An extension of the map such as $\{X \mapsto Y, W \mapsto Z\}$ for W, Z two other symbols is the simultaneous syntactic replacementof X by Y and W by Z. For any two similar signatures Σ, Σ', $\{\Sigma \mapsto \Sigma'\}$ is the subtitution that simultaneously replaces the ith symbol of Σ' with the ith symbol of Σ for every position i of Σ. ■

With a first-order formula ϕ, we can characterise the $\preceq$-minimal elements satisfying ϕ with a 2nd-order formula $\text{↧}\phi$:

Definition (↧ as a 2nd-order formula)**.** If a signature Σ has been partitioned such that $\mathbf{P}_{\text{min}}, \mathbf{P}_{\text{fix}}, \ldots$ are defined and ϕ is a Σ-formula, then:

$$
\begin{aligned}
\text{↧}\phi &:= \phi \wedge \\
&\quad \forall D, \mathbf{M}, \mathbf{V}, \mathbf{f}. \\
&\quad \big(\mathrm{dom}(D, \mathbf{f}) \wedge \mathbf{M} \subseteq \mathbf{P}_{\text{min}} \wedge \phi[D]\{\mathbf{P}_{\text{min}} \mapsto \mathbf{M}, \mathbf{P}_{\text{var}} \mapsto \mathbf{V}, \mathbf{f}_{\text{var}} \mapsto \mathbf{f}\}\big) && (7.1) \\
&\quad \to \big(\mathbf{P}_{\text{min}} \subseteq \mathbf{M} \wedge \forall x. D(x)\big) && (7.2)
\end{aligned}
$$

The right conjunct requires that any structure $\preceq$-smaller than the "current" one is not strictly so – and therefore, equal to the current structure.

It does so by quantifying over possible subdomains (D) as well as minimised and varying predicates & functions ($\mathbf{M}, \mathbf{V}, \mathbf{f}$).

The left-hand side of the implication (line (7.1)) introduces the assumption that D is a subdomain; that $\mathbf{M}$ is indeed smaller than the minimised predicates $\mathbf{P}_{\text{min}}$, and that ϕ is true in this subdomain with the new minimised and varying predicates & functions.

The right-hand side of the implication (line (7.2)) says that the subdomain actually covers the current domain, and that $\mathbf{M} = \mathbf{P}_{\text{min}}$.

- The quantified symbols are D, unary relational; $\mathbf{M}$ is similar to $\mathbf{P}_{\text{min}}$, $\mathbf{V}$ is similar to $\mathbf{P}_{\text{var}}$, and $\mathbf{f}$ is similar to $\mathbf{f}_{\text{var}}$.
- $\phi[D]$ is ϕ where every quantifier has been simultaneously guarded by D. For any atomic formula A, any first-order formulas ψ, ψ', any $\oplus \in \{\wedge, \vee\}$:

$$
\begin{array}{ccc}
A[D] := A & (\neg\psi)[D] := \neg\psi[D] & (\psi \oplus \psi')[D] := \psi[D] \oplus \psi'[D] \\
(\forall x.\ \psi)[D] := \forall x.\ D(x) \to \psi[D] & & (\exists x.\ \psi)[D] := \exists x.\ D(x) \wedge \psi[D]
\end{array}
$$

  for any variable x, any formula ψ.

- dom specifies that D behaves like a nonempety domain that covers $\mathbf{P}_{\text{fix}}$ and the interpretation of the variables:

$$
\mathrm{dom}(D, \mathbf{f}) := \mathbf{f}_{\text{restr}}(D) \subseteq D \wedge \mathbf{f}(D) \subseteq D \wedge \mathbf{P}_{\text{fix}} \subseteq D \wedge \exists x. D(x) \wedge \bigwedge_{x \in \langle\phi\rangle} D(x)
$$

  – For any tuple $\mathbf{g}$ of function symbols, and unary predicate D, $\mathbf{g}(\mathbf{D}) \subseteq D$ means that D is $\mathbf{g}$-closed:

$$
\mathbf{g}(D) \subseteq D := \bigwedge_{f \in \mathbf{g}} \forall \mathbf{x}_f.\ \left(\bigwedge_{x \in \mathbf{x}_f} D(x)\right) \to D(f(\mathbf{x}_f))
$$

where $\mathbf{x}_f$ is a tuple of size the arity of f. Note that whenever f is a constant (i.e. of arity 0), the associated conjunct reduces to $D(f)$.

We do not write the proof of correspondence between $\mathbf{g}(D) \subseteq D$ as a logical specification and the notion in metalanguage.

– For any tuple $\mathbf{P}$ of predicate symbols, and unary predicate D, $\mathbf{P} \subseteq D$ means that any element mentioned in some $S \in \mathbf{P}$ is also in D:

$$\mathbf{P} \subseteq D := \bigwedge_{S \in \mathbf{P}} \forall \mathbf{x}_S.\ S(\mathbf{x}_S) \to \bigwedge_{x \in \mathbf{x}_S} D(x)$$

where $\mathbf{x}_P$ is a tuple of size the arity of P.

We do not write the proof of correspondence between $\mathbf{P} \subseteq D$ as a logical specification and the notion in metalanguage.

– Finally, for any two tuples of predicate symbols $\mathbf{P}, \mathbf{Q}$ of same size and pointwise arity, $\mathbf{P} \subseteq \mathbf{Q}$ means that $\mathbf{P}$ is product-wise included in $\mathbf{Q}$. First, some notation : for a symbol S at position i in $\mathbf{P}$, Q_S is the symbol at position i in $\mathbf{Q}$. By assumption it has the same arity.

$$\mathbf{P} \subseteq \mathbf{Q} := \bigwedge_{S \in \mathbf{P}} \forall \mathbf{x}_S.\ S(\mathbf{x}_S) \to Q_S(\mathbf{x}_S)$$

where $\mathbf{x}_S$ is a tuple of size the arity of S.

We do not write the proof of correspondence between $\mathbf{P} \subseteq \mathbf{Q}$ as a logical specification and the notion in metalanguage.

■

Notation. In the sequel, note that if an interpretation maps symbols of the signature, the interpretation overrides the default meaning of the symbols. For instance, in the signature (P) where P is a unary relational symbol and with the model $\mathfrak{A} := (\mathrm{dom}(\mathfrak{A}) := \{a\}, [\![P]\!]_{\mathfrak{A}} = \{a\})$, consider:

$$\mathfrak{A} \models P(a)$$
$$\mathfrak{A}, P \mapsto \emptyset \not\models P(a)$$

■

Our goal is to prove that a structure $\mathfrak{A}$ is $\preceq$-minimal among models of ϕ iff $\mathfrak{A} \models \downarrow\!\phi$. The following technical lemma is used in *both directions*. It shows that for two structures $\mathfrak{A} \trianglelefteq \mathfrak{B}$, $\mathfrak{A}$ is a model of ϕ iff $\mathfrak{B}$ satisfies a version of ϕ where every quantification is relativised to $\mathrm{dom}(\mathfrak{A})$ and every predicate/function is interpreted as in $\mathfrak{A}$.

Lemma 7.1.1. If $\mathfrak{A}, \mathfrak{B} : \Sigma$, ϕ is a first-order formula, $\mathfrak{A} \preceq \mathfrak{B}$, $\mathbf{M}, \mathbf{V}', \mathbf{g}$ are respectively similar to $\mathbf{P}_{\mathrm{min}}, \mathbf{P}_{\mathrm{var}}, \mathbf{f}_{\mathrm{var}}$, and

$$[\![\mathbf{P}_{\mathrm{min}}]\!]_{\mathfrak{A}} = \mathbf{M}' \restriction \mathrm{dom}(\mathfrak{A}) \qquad [\![\mathbf{P}_{\mathrm{var}}]\!]_{\mathfrak{A}} = \mathbf{V}' \restriction \mathrm{dom}(\mathfrak{A}) \qquad [\![\mathbf{f}_{\mathrm{var}}]\!]_{\mathfrak{A}} = \mathbf{g} \restriction \mathrm{dom}(\mathfrak{A})$$
$$\mu := D \mapsto \mathrm{dom}(\mathfrak{A}), \mathbf{P}_{\mathrm{min}} \mapsto \mathbf{M}', \mathbf{P}_{\mathrm{var}} \mapsto \mathbf{V}', \mathbf{f}_{\mathrm{var}} \mapsto \mathbf{g}$$

then for all $\nu : \langle\phi\rangle \to \mathrm{dom}(\mathfrak{A})$:

$$\mathfrak{A}, \nu \models \phi \text{ iff } \mathfrak{B}, \nu, \mu \models \phi[D]$$

Proof. First, we show that for any term t, $[\![t]\!]_{\mathfrak{A},\nu} = [\![t]\!]_{\mathfrak{B},\nu,\mathbf{f}_{\mathrm{var}}\mapsto \mathbf{g}} \in \mathrm{dom}(\mathfrak{A})$.

- If $t = x$, by $\mathrm{Im}(\nu) \subseteq \mathrm{dom}(\mathfrak{A})$ and $\mathrm{dom}(\mathfrak{A}) \subseteq \mathrm{dom}(\mathfrak{B})$ we are done.
- If $t = f(t_1, ..., t_n)$ with $n \geqslant 0$ and $f \in \mathbf{f}_{\mathrm{restr}}$, using the induction we just need $f(\mathfrak{A}) = f(\mathfrak{B}) \restriction \mathrm{dom}(\mathfrak{A})$, which is given by $[\![\mathbf{f}_{\mathrm{restr}}]\!]_{\mathfrak{B}} \restriction \mathrm{dom}(\mathfrak{A}) = [\![\mathbf{f}_{\mathrm{restr}}]\!]_{\mathfrak{A}}$.
- If $t = f(t_1, ..., t_n)$ with $n \geqslant 0$ and $f \in \mathbf{f}_{\mathrm{var}}$, using the induction we just need $[\![f]\!]_{\mathfrak{A}} = [\![f]\!]_{\mathbf{f}_{\mathrm{var}}\mapsto \mathbf{g}} \restriction \mathrm{dom}(\mathfrak{A})$, which is given by $[\![\mathbf{f}_{\mathrm{var}}]\!]_{\mathfrak{A}} = \mathbf{g} \restriction \mathrm{dom}(\mathfrak{A})$.

Now by induction on ϕ:

- (Equality) $t_1 = t_2$, since for any t, $[\![t]\!]_{\mathfrak{A},\nu} = [\![t]\!]_{\mathfrak{B},\mathbf{f}_{\mathrm{var}}\mapsto \mathbf{g}}$ we are done.
- (Other atomics) $S(\mathbf{t})$, with $\mathbf{t}$ a tuple of terms. By the previous fact, we need $\mathbf{P} = \mathbf{Q} \restriction \mathrm{dom}(\mathfrak{A})$ for $(\mathbf{P}, \mathbf{Q}) \in \{([\![\mathbf{P}_{\mathrm{min}}]\!]_{\mathfrak{A}}, \mathbf{M}'), ([\![\mathbf{P}_{\mathrm{var}}]\!]_{\mathfrak{A}}, \mathbf{V}'), ([\![\mathbf{P}_{\mathrm{restr}}]\!]_{\mathfrak{A}}, [\![\mathbf{P}_{\mathrm{restr}}]\!]_{\mathfrak{B}}), ([\![\mathbf{P}_{\mathrm{fix}}]\!]_{\mathfrak{A}}, [\![\mathbf{P}_{\mathrm{fix}}]\!]_{\mathfrak{B}})\}$. This is all by hypothesis.
- (Bool) The case of propositional connectives is trivial.
- $(\forall, \exists)$ For quantifiers, thanks to negation we treat $\forall$ only wlog. If $\mathfrak{A}, \nu \models \forall x.\ \psi$ then for any $a \in \mathrm{dom}(\mathfrak{A})$, $\mathfrak{A}, \nu, x \mapsto a \models \psi$ and the induction gives us $\mathfrak{B}, \nu, x \mapsto a, \mu \models \psi[D]$. Since $\mu(D) = \mathrm{dom}(\mathfrak{A})$, we get $\mathfrak{B}, \nu, \mu \models \forall x.\ D(x) \to \psi[D]$, i.e. $\mathfrak{B}, \nu, \mu \models (\forall x.\ \psi)[D]$.

  The other direction works the same.

□

The main lemma of this section shows the semantics-syntax correspondence of $\downarrow$ and $\preceq$. The key lemma is lemma 7.1.1 in both directions. From syntax to semantics, given $\mathfrak{A}, \mu \models \downarrow\phi$, the proof uses the key lemma to put the "innards" of a would-be smaller strictly model $\mathfrak{B}$ as instance of the second-order quantifiers of $\downarrow\phi$, which leads to $\mathfrak{A} \preceq \mathfrak{B}$. From semantics to syntax, we do the same inverse, taking all instance sof the second-order quantifications and showing, through minimality and the key lemma, that they verify the inside the $\downarrow\phi$.

Lemma 7.1.2. If ϕ is a first-order formula, the models of $\downarrow\phi$ are the $\preceq$-minimal models of ϕ.

Proof.

($\Rightarrow$) Suppose $\mathfrak{A}, \mu \models \downarrow\phi$ and let $\mathfrak{B}, \mu \models \phi$, $\mathfrak{B} \preceq \mathfrak{A}$. We want $\mathfrak{A} \preceq \mathfrak{B}$. By definition of $\downarrow\phi$, we get:

$$\begin{aligned}
&\mathfrak{A}, \mu,\ D \mapsto \mathrm{dom}(\mathfrak{B}),\ \mathbf{M} \mapsto [\![\mathbf{P}_{\mathrm{min}}]\!]_{\mathfrak{B}},\ \mathbf{V} \mapsto [\![\mathbf{P}_{\mathrm{var}}]\!]_{\mathfrak{B}},\ \mathbf{f} \mapsto [\![\mathbf{f}_{\mathrm{var}}]\!]_{\mathfrak{B}}\\
&\quad \models \mathrm{dom}(D, \mathbf{f}) \wedge \mathbf{M} \subseteq \mathbf{P}_{\mathrm{min}} \wedge \phi[D]\{\mathbf{P}_{\mathrm{min}} \mapsto \mathbf{M}, \mathbf{P}_{\mathrm{var}} \mapsto \mathbf{V}, \mathbf{f}_{\mathrm{var}} \mapsto \mathbf{f}\}\\
&\qquad \to \big(\mathbf{P}_{\mathrm{min}} \subseteq \mathbf{M} \wedge \forall x. D(x)\big)
\end{aligned}$$

We show that the left-hand side of the implication holds.

The second conjunct ($\mathbf{M} \subseteq \mathbf{P}_{\mathrm{min}}$) is given by $\mathfrak{B} \preceq \mathfrak{A}$. The third conjunct ($\phi[D]\{\ldots\}$) is given by lemma 7.1.1, using $\mathfrak{B}, \mu \models \phi$ and $\mathfrak{B} \preceq \mathfrak{A}$ (note that we apply the lemma with $\mathbf{M}' := [\![\mathbf{P}_{\mathrm{min}}]\!]_{\mathfrak{A}}$, $\mathbf{V}' := [\![\mathbf{P}_{\mathrm{var}}]\!]_{\mathfrak{A}}$, and $\mathbf{g} := [\![\mathbf{f}_{\mathrm{var}}]\!]_{\mathfrak{A}}$).

Now for the first conjunct ($\mathrm{dom}(D, \mathbf{f})$).

We need $\mathfrak{A}, D \mapsto \mathrm{dom}(\mathfrak{B}) \models \exists x.\ D(x)$. Since $\mathrm{dom}(\mathfrak{B})$ is non-empty and $\mathrm{dom}(\mathfrak{B}) \subseteq \mathrm{dom}(\mathfrak{A})$ by $\preceq$, there is some $a \in \mathrm{dom}(\mathfrak{A}) \cap \mathrm{dom}(\mathfrak{B})$.

We also need $\mathrm{dom}(\mathfrak{B})$ to be $[\![\mathbf{f}_{\text{restr}}]\!]_{\mathfrak{A}}$-closed and $[\![\mathbf{f}_{\text{var}}]\!]_{\mathfrak{B}}$-closed. Since $[\![\mathbf{f}_{\text{restr}}]\!]_{\mathfrak{B}} = [\![\mathbf{f}_{\text{restr}}]\!]_{\mathfrak{A}} \restriction \mathrm{dom}(\mathfrak{B})$ (by $\preceq$), $\mathrm{dom}(\mathfrak{B})$ is $[\![\mathbf{f}_{\text{restr}}]\!]_{\mathfrak{A}}$-closed. The other is by definition.

We need $[\![\mathbf{P}_{\text{fix}}]\!]_{\mathfrak{A}} \subseteq D$. Since $[\![\mathbf{P}_{\text{fix}}]\!]_{\mathfrak{B}} = [\![\mathbf{P}_{\text{fix}}]\!]_{\mathfrak{A}}$ (by $\preceq$) and we have $[\![\mathbf{P}_{\text{fix}}]\!]_{\mathfrak{B}} \subseteq \mathrm{dom}(\mathfrak{B})$ by definition, we get $[\![\mathbf{P}_{\text{fix}}]\!]_{\mathfrak{A}} \subseteq \mathrm{dom}(\mathfrak{B})$.

The left-hand side of the implication being proven, we use the right-hand side. We want $\mathfrak{A} \preceq \mathfrak{B}$. We have $[\![\mathbf{P}_{\text{min}}]\!]_{\mathfrak{A}} \subseteq [\![\mathbf{P}_{\text{min}}]\!]_{\mathfrak{B}}$ and $\mathrm{dom}(\mathfrak{A}) \subseteq \mathrm{dom}(\mathfrak{B})$, so $\mathfrak{A} \preceq \mathfrak{B}$ indeed.

(⇐) Suppose $\mathfrak{A}$ is $\preceq$-minimal among models of ϕ with interpretation μ for its free variables. We show $\mathfrak{A}, \mu \models \rfloor\phi$. By assumption $\mathfrak{A}, \mu \models \phi$. Now take any predicate tuples $D', \mathbf{M}', \mathbf{V}'$ (with size and arity matching the symbols in $D, \mathbf{M}, \mathbf{V}$) and functions $\mathbf{f}'$ (with size and arity matching the symbols in $\mathbf{f}$) such that

$$\begin{aligned}&\mathfrak{A}, \mu, D \mapsto D', \mathbf{M} \mapsto \mathbf{M}', \mathbf{V} \mapsto \mathbf{V}', \mathbf{f} \mapsto \mathbf{f}' \\ &\quad \models \mathrm{dom}(D, \mathbf{f}) \wedge \mathbf{M} \subseteq \mathbf{P}_{\text{min}} \wedge \phi[D]\{\mathbf{P}_{\text{min}} \mapsto \mathbf{M}, \mathbf{P}_{\text{var}} \mapsto \mathbf{V}, \mathbf{f}_{\text{var}} \mapsto \mathbf{f}\}\end{aligned}$$

Let $\mathfrak{B} : \Sigma$ be such that

$$\begin{array}{lll} \mathrm{dom}(\mathfrak{B}) = D' & [\![\mathbf{P}_{\text{min}}]\!]_{\mathfrak{B}} = \mathbf{M}' \restriction D' & [\![\mathbf{P}_{\text{restr}}]\!]_{\mathfrak{B}} = [\![\mathbf{P}_{\text{restr}}]\!]_{\mathfrak{A}} \restriction \mathrm{dom}(\mathfrak{B}) \\ [\![\mathbf{f}_{\text{restr}}]\!]_{\mathfrak{B}} = [\![\mathbf{f}_{\text{restr}}]\!]_{\mathfrak{A}} \restriction \mathrm{dom}(\mathfrak{B}) & [\![\mathbf{P}_{\text{var}}]\!]_{\mathfrak{B}} = \mathbf{V}' \restriction \mathrm{dom}(\mathfrak{B}) & [\![\mathbf{f}_{\text{var}}]\!]_{\mathfrak{B}} = \mathbf{f}' \restriction \mathrm{dom}(\mathfrak{B}) \\ [\![\mathbf{P}_{\text{fix}}]\!]_{\mathfrak{B}} = [\![\mathbf{P}_{\text{fix}}]\!]_{\mathfrak{A}} & & \end{array}$$

We must check that $\mathfrak{B}$ is well-defined.

Using the dom subformula, $\mathfrak{B}$ is not the empty model since there is at least one element in $\mathrm{dom}(\mathfrak{B})$, $\mathrm{dom}(\mathfrak{B})$ is $[\![\mathbf{f}_{\text{restr}}]\!]_{\mathfrak{A}}$-closed, $\mathrm{dom}(\mathfrak{B})$ is $[\![\mathbf{f}_{\text{var}}]\!]_{\mathfrak{B}}$-closed, and we also get $[\![\mathbf{P}_{\text{fix}}]\!]_{\mathfrak{B}} \subseteq \mathrm{dom}(\mathfrak{B})$.

Since $[\![\mathbf{P}_{\text{min}}]\!]_{\mathfrak{B}} \subseteq \mathbf{M}' \subseteq [\![\mathbf{P}_{\text{min}}]\!]_{\mathfrak{A}}$, we get $\mathfrak{B} \preceq \mathfrak{A}$. Using $\bigwedge_{x \in \langle\phi\rangle} D(x)$, we can now apply lemma 7.1.1 to get $\mathfrak{B}, \mu \models \phi$. By minimality of $\mathfrak{A}$ among models of ϕ, this implies $\mathfrak{A} \preceq \mathfrak{B}$, i.e. $[\![\mathbf{P}_{\text{min}}]\!]_{\mathfrak{A}} \subseteq [\![\mathbf{P}_{\text{min}}]\!]_{\mathfrak{B}} = \mathbf{M}' \restriction D' \subseteq \mathbf{M}'$ and $\mathrm{dom}(\mathfrak{A}) \subseteq \mathrm{dom}(\mathfrak{B}) = D'$. Which gives us the desired conclusion :

$$\begin{aligned}&\mathfrak{A}, \mu,\ D \mapsto D',\ \mathbf{M} \mapsto \mathbf{M}' \\ &\quad \models \mathbf{P}_{\text{min}} \subseteq \mathbf{M} \wedge \forall x. D(x)\end{aligned}$$

□

Following [Etherington, 1986] (p.142, Theorem 6.3) and [De Kleer and Konolige, 1989], it is easy to see that, in addition to relational symbols, formulas that use fixed and varying symbols can also be minimised, that is: $\mathbf{P}_{\text{min}}$ can also contain formulas that do not use other symbols from $\mathbf{P}_{\text{min}}$. To sketch the construction: take a signature Σ partitioned into $\mathbf{P}_{\text{min}}$, $\mathbf{f}_{\text{restr}}$, etc, such that an order $\preceq$ is defined. Given an FO Σ-formula θ that uses only symbols in $(=, \mathrm{P}_{\text{fix}}, \mathrm{P}_{\text{var}})$, we define an order $\preceq_\theta$: $\mathfrak{A} \preceq_\theta \mathfrak{B}$ if $\mathfrak{A} \preceq \mathfrak{B}$ and $[\![\theta]\!]_{\mathfrak{A}} \subseteq [\![\theta]\!]_{\mathfrak{B}}$. Extend the signature Σ to $\Sigma' \coloneqq (\Sigma, P)$, where P is a fresh relational symbol of arity $|\langle\theta\rangle|$. P is added to the tuple $\mathbf{P}_{\text{min}}$ of Σ'. This gives rise to an order $\preceq'$ on Σ'-structures. The function f from Σ-structures to Σ'-structures which just extends any $\mathfrak{A} : \Sigma$ to Σ' by adding $[\![P]\!]_{f(\mathfrak{A})} \coloneqq [\![\theta]\!]_{\mathfrak{A}}$ is a bijection between Σ-structures and Σ'-models of $(\forall \mathbf{x}.\ P(\mathbf{x}) \leftrightarrow \theta(\mathbf{x}))$. Moreover *a)* $\mathfrak{A} \preceq_\theta \mathfrak{B}$ iff $f(\mathfrak{A}) \preceq' f(\mathfrak{B})$ and *b)* for any Σ'-formula F and $\mu : \langle F\rangle \to \mathrm{dom}(\mathfrak{A})$, $f(\mathfrak{A}), \mu \models F$ iff $\mathfrak{A}, \mu \models F'$ where F' is F with all atoms of the form $P(\mathbf{x})$ replaced by $\theta(\mathbf{x})$. This applies in particular when $F = \rfloor\phi$ for some FO, Σ'-formula ϕ, so the definition for circumscribed formulas page 93 admits having formulas (and not just symbols) in $\mathbf{P}_{\text{min}}$.

7.2. $\trianglelefteq$-minimality in 2nd-order logic

In this section we make the simple observation (using the previous section) that the minimisation operator $\downarrow$ can be expressed syntactically as an instance of unified circumscription. Let $\Delta\mathbf{Dyn}$ be the tuple of formulas of the form $\Delta A(\mathbf{x}) = \neg(A(\mathbf{x}) \leftrightarrow A^\star(\mathbf{x}))$ for $A \in \mathbf{Dyn}$ and $\mathbf{x}$ a tuple of variables of size the arity of A. We define an order $\preceq$ using the following mapping:

$$\mathbf{P}_{\text{min}} = \Delta\mathbf{Dyn} \qquad \mathbf{P}_{\text{restr}} = \mathbf{Stat} \setminus \mathbf{Stat}_{\text{fun}} \qquad \mathbf{P}_{\text{var}} = \mathbf{Dyn}^\star$$
$$\mathbf{P}_{\text{fix}} = \mathbf{Dyn} \qquad \mathbf{f}_{\text{restr}} = \mathbf{Stat}_{\text{fun}} \qquad \mathbf{f}_{\text{var}} = \emptyset$$

This defines a circumscription order $\preceq$ on Θ-structures.

Lemma 7.2.1. If $\mathfrak{A}, \mathfrak{B} : \Theta$, $\mathfrak{A} \preceq \mathfrak{B}$ iff $\mathfrak{A} \trianglelefteq \mathfrak{B}$

Proof. Cf. definitions for $\trianglelefteq$ (p. 34) and $\preceq$ (p. 92): (a) iff (4), (b) iff (3), (c) and (d) iff (2), and finally (e) iff (1). □

We restate the formula for $\downharpoonleft$, adapted to the signature Θ and the mapping to $\mathbf{P}_{\text{min}}$, etc:

$$\begin{aligned}
\psi_1(D, \mathbf{Dyn}') &:= \mathbf{g}(D) \subseteq D \land \mathbf{Dyn} \subseteq D \land (\exists x.\ D(x)) \land \bigwedge_{x \in \langle\phi\rangle} D(x) \\
&\quad \land \Delta\mathbf{Dyn}' \subseteq \Delta\mathbf{Dyn} \land \phi[D]\{\mathbf{Dyn}^\star \mapsto \mathbf{Dyn}'\} \\
\psi_2(D, \mathbf{Dyn}') &:= \Delta\mathbf{Dyn} \subseteq \Delta\mathbf{Dyn}' \land \forall x.\ D(x) \\
\downharpoonleft\phi &= \phi\ \land \forall D, \mathbf{Dyn}'.\ \psi_1(D, \mathbf{Dyn}') \to \psi_2(D, \mathbf{Dyn}')
\end{aligned}$$

Where $\mathbf{Dyn}'$ is a tuple similar to $\mathbf{Dyn}^\star$ and $\Delta\mathbf{Dyn}'$ is $\Delta\mathbf{Dyn}$ with every predicate in $\mathbf{Dyn}^\star$ replaced by its counterpart in $\mathbf{Dyn}'$. For instance, $\Delta\mathsf{Has}' = \neg(\mathsf{Has} \leftrightarrow \mathsf{Has}')$ (variables omitted).

We have only performed simple syntactic renaming and removed some second-order quantification since there are no variable functions ($\mathbf{f}_{\text{var}}$).

By lemma 7.2.1 we have the following equivalence:

$$\downarrow\phi \equiv \phi\ \land \forall D, \mathbf{Dyn}'.\ \psi_1(D, \mathbf{Dyn}') \to \psi_2(D, \mathbf{Dyn}')$$

7.3. 2nd-order quantifier elimination

In this section we show when we can eliminate $\downarrow$ from FO[$\downarrow$]-formulas.We start from the observation that quantifying on subdomains is unnecessary when all the models of a formula are supported. Then, we define the *boundedness* condition on formulas, which restrict the amount of changes that can be present in their minimal models. Note that as seen earlier, the minimal models of any $\models_{\trianglelefteq}$-provable formula contain a bounded amount of changes. Modulo that boundedness property on a formula ϕ, all 2nd-order quantifiers can be removed from the 2nd-order formula equivalent to $\downarrow\phi$. During this quantifier elimination, we also notice that first-order quantifiers can be simplified modulo the theory of n-BFLs and $\models_{\rho}$-provability. As a result, every $\models_{\rho}$-provable, minimised formula is equivalent to a formula in $\exists^*\forall^*$ and to a formula in $\forall^*\exists^*$.

7.3.1. Removing domain quantification

In the sequel we will substitute relational symbols with formulas of the same arity. For instance, here we pass a formula $\theta(x)$ with 1 free variable as an argument to $\psi_1(\cdot,\cdot)$:

$$\psi_2(\theta, \mathbf{Dyn}') = \Delta\mathbf{Dyn}' \subseteq \Delta\mathbf{Dyn}' \wedge \forall x.\ \theta(x)$$

Definition. To keep things syntactically simple, we define a context-dependent formula. In any context where a symbol named Has' is defined, let $\mathit{Supp}'(x) := \mathsf{Has}(x) \vee \mathsf{Has}'(x)$. ■

It is possible to simplify a circumscribed formula $\wr\phi$ if we assume all models are supported, i.e. that the pre and post-states cover the entire domain. This is what the next lemma shows:

Lemma 7.3.1. If $\phi \vDash \mathit{Support}$ then:

$$\wr\phi \equiv \phi \wedge \forall \mathbf{Dyn}'.\ \psi_1(\mathit{Supp}', \mathbf{Dyn}') \rightarrow \psi_2(\mathit{Supp}', \mathbf{Dyn}')$$

Proof. First, note the difference with $\wr\phi$ as described in the beginning of this section: the second-order quantification $\forall D$ is gone, and all atoms of the form $D(x)$ for x any variable have been replaced by $\mathit{Supp}'(x)$. This is well-defined since the replacement occurs within a quantification on $\mathbf{Dyn}'$ and there is a symbol $\mathsf{Has}' \in \mathbf{Dyn}'$.

($\Rightarrow$) Assume $\mathfrak{A}, \mu \vDash \forall D, \mathbf{Dyn}'.\ \psi_1(D, \mathbf{Dyn}') \rightarrow \psi_2(D, \mathbf{Dyn}')$. For any predicate tuple $\mathbf{M}$ such that $\mathfrak{A}, \mu, \mathbf{Dyn}' \mapsto \mathbf{M} \vDash \psi_1(\mathit{Supp}', \mathbf{Dyn}')$, we need $\mathfrak{A}, \mu, \mathbf{Dyn}' \mapsto \mathbf{M} \vDash \Delta\mathbf{Dyn} \subseteq \Delta\mathbf{Dyn}' \wedge \mathit{Support}$.

With H the image of Has' in $\mathbf{Dyn}' \mapsto \mathbf{M}$, it suffices to show $\mathfrak{A}, \mu, D \mapsto ([\![\mathsf{Has}]\!]_{\mathfrak{A}} \cup H), \mathbf{Dyn}' \mapsto \mathbf{M} \vDash \psi_1(D, \mathbf{Dyn}')$. Indeed, every conjunct of ψ_1 is trivially satisfied by assumption, so we are done.

($\Leftarrow$) For the other direction, we assume $\mathfrak{A}, \mu \vDash \forall \mathbf{Dyn}'.\ \psi_1(\mathit{Supp}', \mathbf{Dyn}') \rightarrow \psi_2(\mathit{Supp}', \mathbf{Dyn}')$.

Take any subset T of $\mathrm{dom}(\mathfrak{A})$ and any appropriately-sized tuple $\mathbf{M}$ of tuples of subsets of $\mathrm{dom}(\mathfrak{A})$ such that $\mathfrak{A}, \mu, D \mapsto T, \mathbf{Dyn}' \mapsto \mathbf{M} \vDash \psi_1(D, \mathbf{Dyn}')$.

Wlog, assume Has' is the first symbol of $\mathbf{Dyn}'$, and let H be the first component of $\mathbf{M}$. Let $\mathbf{M}_{HT}$ be $\mathbf{M}$ with its first component replaced with $H \cap T$.

We have $[\![\mathsf{Has}]\!]_{\mathfrak{A}} \subseteq T$ by ψ_1. So $[\![\mathsf{Has}]\!]_{\mathfrak{A}} \cup (H \cap T) \subseteq T$. By hypothesis on ϕ, we have $T \subseteq [\![\mathsf{Has}]\!]_{\mathfrak{A}} \cup (H \cap T)$. So we get $T = [\![\mathsf{Has}]\!]_{\mathfrak{A}} \cup (H \cap T)$.

- We first show $\mathfrak{A}, \mu, \mathsf{Has}' \mapsto \mathbf{M}_{HT} \vDash \psi_1(\mathit{Supp}', \mathbf{Dyn}')$.

  Remember that $T = [\![\mathsf{Has}]\!]_{\mathfrak{A}} \cup (H \cap T)$. By assumption we do have $[\![\mathfrak{g}]\!]_{\mathfrak{A}}(T) \subseteq T$ and $[\![\mathbf{Dyn}]\!]_{\mathfrak{A}} \subseteq T$ as well as T nonempty.

  To show $\Delta\mathbf{M}_{HT} \subseteq [\![\Delta\mathbf{Dyn}]\!]_{\mathfrak{A}}$, remember that $\Delta\mathbf{M} \subseteq [\![\Delta\mathbf{Dyn}]\!]_{\mathfrak{A}}$. Suppose there is some e which is in $[\![\mathsf{Has}]\!]_{\mathfrak{A}}\Delta(H \cap T)$ but not in $[\![\mathsf{Has}]\!]_{\mathfrak{A}}\Delta H$. Since $H \cap T \subseteq H$, $e \in T$ is impossible. So $e \in [\![\mathsf{Has}]\!]_{\mathfrak{A}}$. But by hypothesis, $[\![\mathsf{Has}]\!]_{\mathfrak{A}} \subseteq T$, which is absurd.

  Finally, we need $\mathfrak{A}, \mu, \mathbf{Dyn}' \mapsto \mathbf{M}_{HT} \vDash \phi[\mathit{Supp}'](\mathbf{Dyn}, \mathbf{Dyn}', \mathsf{Stat})$. We will use $\mathfrak{A}, \mu, D \mapsto T, \mathbf{Dyn}' \mapsto \mathbf{M} \vDash \phi[D](\mathbf{Dyn}, \mathbf{Dyn}', \mathsf{Stat})$:

  We show that $\mathfrak{A}, \mu, D \mapsto T, \mathbf{Dyn}' \mapsto \mathbf{M} \vDash \phi[D](\mathbf{Dyn}, \mathbf{Dyn}', \mathsf{Stat})$ iff $\mathfrak{A}, \mu, \mathbf{Dyn}' \mapsto \mathbf{M}_{HT} \vDash \phi[\mathit{Supp}'](\mathbf{Dyn}, \mathbf{Dyn}', \mathsf{Stat})$ by induction on ϕ. Note that, from left to right, we can assume that μ goes into T (by ψ_1), and from right to left, that μ goes into $[\![\mathsf{Has}]\!]_{\mathfrak{A}} \cup (H \cap T)$ (again by ψ_1).

  The only nontrivial case is $\phi = \mathsf{Has}^\star(t)$. Then, by induction on t:

  - $t = x$ then either $\mu(x) \in [\![\mathsf{Has}]\!]_{\mathfrak{A}} \subseteq T$ so $\mu(x) \in H$ iff $\mu(x) \in (H \cap T)$; or $\mu(x) \in (H \cap T)$ then we are done.
  - $t = f(t')$, again by $[\![g]\!]_{\mathfrak{A}}(T) \subseteq T$ and induction hypothesis we are done.

- Now that we know $\mathfrak{A}, \mu, \mathbf{Dyn}' \mapsto M_{HT} \models \psi_1(\mathit{Supp}', \mathbf{Dyn}')$, we can use the main assumption to obtain $\mathfrak{A}, \mu, \mathbf{Dyn}' \mapsto M_{HT} \models \Delta\mathbf{Dyn} \subseteq \Delta\mathbf{Dyn}' \wedge \mathit{Support}$. We will deduce $\mathfrak{A}, \mu, D \mapsto T, \mathbf{Dyn}' \mapsto M \models \Delta\mathbf{Dyn} \subseteq \Delta\mathbf{Dyn}' \wedge \forall x.\ D(x)$ from it:

  First, $T = [\![\mathsf{Has}]\!]_{\mathfrak{A}} \cup (H \cap T)$, $[\![\mathsf{Has}]\!]_{\mathfrak{A}} \Delta [\![\mathsf{Has}^\star]\!]_{\mathfrak{A}} \subseteq [\![\mathsf{Has}]\!]_{\mathfrak{A}} \Delta (H \cap T)$ (obtained from $\Delta\mathbf{Dyn} \subseteq \Delta\mathbf{Dyn}'$ with $\mathbf{Dyn}' \mapsto M_{HT}$) and *Support*, yield $T = \mathrm{dom}(\mathfrak{A})$ through simple set-theoretic considerations.

  Let $\Delta M := [\![\mathbf{Dyn}]\!]_{\mathfrak{A}} \Delta M$ and $\Delta M_{HT} := [\![\mathbf{Dyn}]\!]_{\mathfrak{A}} \Delta M_{HT}$.

  By definition of M_{HT}, and since $[\![\Delta\mathbf{Dyn}]\!]_{\mathfrak{A}} \subseteq \Delta M_{HT}$, we may only get $[\![\Delta\mathbf{Dyn}]\!]_{\mathfrak{A}} \not\subseteq \Delta M$ if there is some e in $[\![\Delta\mathsf{Has}]\!]_{\mathfrak{A}}$ (and thus in $[\![\mathsf{Has}]\!]_{\mathfrak{A}} \Delta (H \cap T)$), but not in $[\![\mathsf{Has}]\!]_{\mathfrak{A}} \Delta H$. By simple set-theoretic considerations, we get $e \in [\![\mathsf{Has}]\!]_{\mathfrak{A}}$ and $e \notin T$; the latter is absurd. So $[\![\Delta\mathbf{Dyn}]\!]_{\mathfrak{A}} \subseteq \Delta M$.

$\square$

7.3.2. Removing 2nd-order quantification

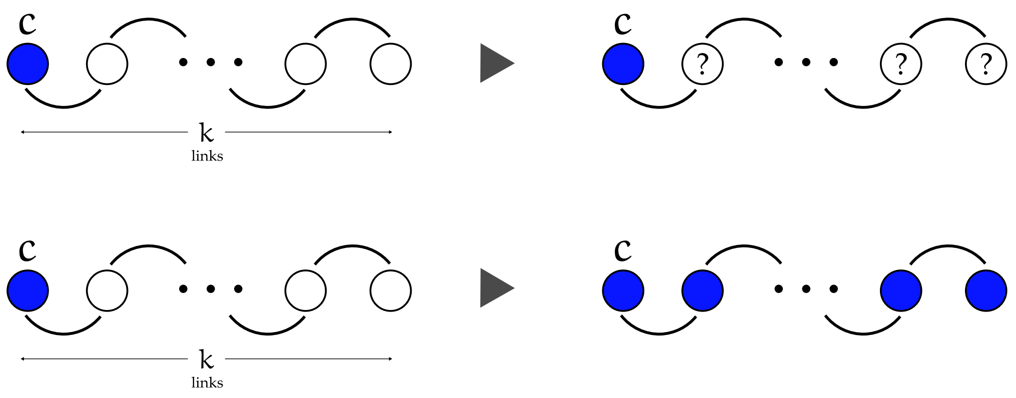


Figure 7.2.: Precondition of $\mathfrak{A}_k$. What must the postcondition be to have $\mathfrak{A}_k \models \downarrow CC$?

Example. As an example of a minimised formula which is not FO-definable, we reprise the example from page 59 (formula reproduced below).

$$\begin{aligned}\downarrow CC := \downarrow(&\mathsf{Blue}(c) \wedge \\ &\forall x.\ (x \neq c \to \neg\mathsf{Blue}(x)) \wedge \\ &\forall y.\ \mathsf{Blue}^\star(x) \to \mathsf{Link}^\star(x, y) \to \mathsf{Blue}^\star(y))\end{aligned}$$

with $\mathsf{Blue} \in \mathbf{Dyn}$. $CC(c)$ specifies *a)* that the interpretation of c is Blue and no other element is blue, and *b)* an element with a $\mathsf{Blue}^\star$ neighbor must be $\mathsf{Blue}^\star$.

Note that the query defined by $\mathsf{Blue}^\star(y) \wedge \downarrow CC$ captures exactly the elements accessible from $[\![c]\!]_{\mathfrak{A}}$.

To sketch a proof of non-definability in FO: all FO-definable queries are *Gaifman-local*, in the sense that for all such queries Q, there is some $d \geqslant 0$ such that (for the case of queries of arity 1)

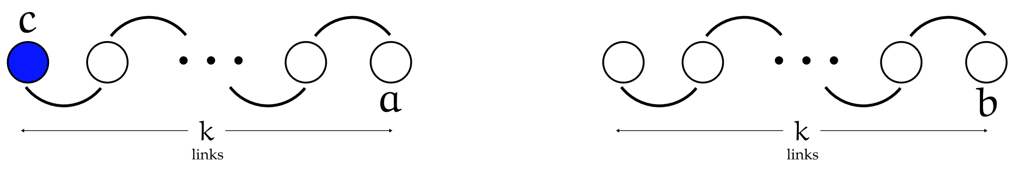


Figure 7.3.: The d-balls around a and b are isomorphic

for all structures $\mathfrak{A}$ and $a, b \in \mathrm{dom}(\mathfrak{A})$, if the d-balls around a and b are isomorphic (with a the image of b in the isomorphism), then $a \in Q(\mathfrak{A})$ iff $b \in Q(\mathfrak{A})$. For more on Gaifman-locality, see [Libkin, 2013], chapter 4. We skip over the fact that Gaifman locality is defined for relational signatures and our signatures are non-relational since functions are trivial to encode with first-order formulas and relational symbols.

Now for any d, take any n-BFL $\mathfrak{A}$ with two connected components of size $k > d$, as in figure 7.3 (we only see the preconditions in that figure). c is at one end of one of the connected components. Let a be the other extremity of c's connected component, and let b be any extremity of the other. The d-balls around a and b are isomorphic, but $\mathfrak{A}, y \mapsto a \models \mathsf{Blue}^\star(y) \wedge \downarrow CC$ and $\mathfrak{A}, y \mapsto b \not\models \mathsf{Blue}^\star(y) \wedge \downarrow CC$ (cf. figure 7.2). So $\mathsf{Blue}^\star(y) \wedge \downarrow CC$ is not FO-definable.

We have allowed the interpretation of $\mathsf{Link}^\star$ to be of degree 2 to make things easier; the same construction works on n-BFLs by using trees of height 1.

■

Example. We will soon show that bounding the amount of changes makes any minimised formula FO-definable. However, there are minimised formulas with unbounded changes that are FO-definable. A very simple example :

$$\mathit{Switch} := \forall x.\ \neg \mathsf{A}(x) \wedge \mathsf{A}^\star(x)$$

Any model $\mathfrak{A}$ of $\downarrow \mathit{Switch}$ contains $|\mathrm{dom}(\mathfrak{A})|$ changes. ■

We give a semantic constraint under which 2nd-order quantification can be removed. The idea is that if, for a formula ϕ, there is a bound on the amount of changes contained in its minimal models, then second-order quantification in $\updownarrow\phi$ can be replaced by a finite enumeration.

Definition. If $\beta : \mathbf{Dyn} \to \mathbb{N} = \mathsf{A} \mapsto \beta_\mathsf{A}$, ϕ is formula, ψ is a sentence, ϕ is *β-bounded* if for every dynamic predicate $\mathsf{A} \in \mathbf{Dyn}$, for any $\mathfrak{A}, \mu \models \psi \wedge \downarrow \phi$, $|[\![\Delta \mathsf{A}]\!]_\mathfrak{A}| \leqslant \beta_\mathsf{A}$. β_A is then a bound for A in ϕ modulo ψ. ■

In other words, among the $\trianglelefteq$-minimal models of a bounded formula ϕ, there is a bound on the number of elements that can change.

We say that a formula is bounded modulo ψ if there exists a β such that the formula is β-bounded modulo ψ. We skip the "modulo" part to mean that ψ is valid.

Definition. If ϕ is β-bounded modulo ψ, define $\mathbf{X}^\beta_\mathsf{A} := x_1, \ldots, x_{\beta_\mathsf{A}}$, a tuple *of tuples* of fresh variables, with $|x_i| = \mathrm{arity}(\mathsf{A})$ for $1 \leqslant i \leqslant \beta_\mathsf{A}$. ■

Definition. If ϕ is β-bounded modulo ψ and $\mathsf{A} \in \mathbf{Dyn}$, define an an associated *bounding formula*, *Bound*(A):

$$\mathit{Bound}(\mathsf{A}) := (\forall x.\ \mathsf{A}^\star(x) \leftrightarrow \mathsf{A}(x)) \vee \mathit{Diff}(\mathsf{A}^\star, \mathsf{A}, \mathbf{X}^\beta_\mathsf{A})$$

where for all relational symbols S, S' of same arity a, for all $\mathbf{X}_k := \mathbf{x}_1, \dots, \mathbf{x}_k$ with $k \geqslant 0$ and such that $|\mathbf{x}_i| = a$ for $1 \leqslant i \leqslant k$:

$$\begin{aligned} \mathit{Diff}(S', S, \mathbf{X}_k) &:= \forall \mathbf{x}.\ S'(\mathbf{x}) \leftrightarrow \mathit{Split}(S, \mathbf{X}_k, \mathbf{x}) \\ \mathit{Split}(S, \mathbf{X}_k, \mathbf{x}) &:= (S(\mathbf{x}) \wedge \mathbf{x} \notin \mathbf{X}_k) \vee (\neg S(\mathbf{x}) \wedge \mathbf{x} \in \mathbf{X}_k) \end{aligned}$$

The quantified $\mathbf{x}$ also has size a. $\mathbf{x} \in \mathbf{X}_k$ is shorthand for $\mathbf{x} = \mathbf{x_1} \vee \dots \vee \mathbf{x} = \mathbf{x}_k$. Again, note that each $\mathbf{x}_i$ is a tuple of variables (of size a) and that $\mathbf{X}_k$ is a k-sized tuple *of tuples* of variables. ■

A bounding formula for A declares that there are at most β_A tuples such that every tuple involved in a change (i.e. such that it is true in A and false in $A^\star$, or the reverse) is also one of the β_A tuples, and that those are the only such tuples.

More precisely:

- *Diff* takes predicates S' and S as well as tuples $\mathbf{X}_k = \mathbf{x}_1, \dots, \mathbf{x}_k$. It states that a tuple $\boldsymbol{a}$ of elements (in some structure $\mathfrak{A}$ satisfying the formula) is in $[\![S']\!]_{\mathfrak{A}} \Delta [\![S]\!]_{\mathfrak{A}}$ iff it is equal to the interpretation of some $\mathbf{x}_i$.

- *Bound* assumes the existence of some bound β_A for its argument A and specifies that either $A^\star$ and A are equal, or $\mathit{Diff}(A^\star, A, \mathbf{X}^\beta_A)$ holds. The first part ("$A^\star$ iff A") is necessary because while $\mathbf{X}^\beta_A$ must be interpreted by some actual tuples, β_A is an overapproximation of the size of the difference between A and $A^\star$; so it's necessary to explicitly allow for no difference at all.

Lemma 7.3.2. If ϕ is β-bounded modulo ψ, then for any $A \in \mathbf{Dyn}$, if $\mathfrak{A}, \mu \vDash \psi \wedge \downarrow\phi$ then there is $\mathbf{M}_A$ such that $\mathfrak{A}, \mathbf{X}^\beta_A \mapsto \mathbf{M}_A \vDash \mathit{Bound}(A)$.

Proof. By minimality of $\mathfrak{A}$ and boundedness of ϕ modulo ψ, there are $k \leqslant \beta_A$ tuples in $[\![\Delta A]\!]_{\mathfrak{A}}$. If $k = 0$ then $\mathfrak{A} \vDash \forall x.\ A(x) \leftrightarrow A^\star(x)$. Otherwise let $\{\mathbf{p}_1, \dots, \mathbf{p}_k\} = [\![\Delta A]\!]_{\mathfrak{A}}$. Define $\mathbf{M}_A := \mathbf{u}_1, \dots, \mathbf{u}_{\beta_A}$ where $\mathbf{u}_i := \mathbf{p}_{i \bmod k}$. We must show:

$$\mathfrak{A} \vDash \forall \mathbf{x}.\ A^\star(\mathbf{x}) \leftrightarrow ((A(\mathbf{x}) \wedge (\bigwedge_i \mathbf{x} \neq \mathbf{u}_i)) \vee (\neg A(\mathbf{x}) \wedge (\bigvee_i \mathbf{x} = \mathbf{u}_i)))$$

(This is just an inlining of the definition of *Bound*). If $\boldsymbol{e} \in [\![A^\star]\!]_{\mathfrak{A}}$, then either $\boldsymbol{e} \in [\![\Delta A]\!]_{\mathfrak{A}}$, so $\boldsymbol{e} \notin [\![A]\!]_{\mathfrak{A}}$ and there is $1 \leqslant j \leqslant k$ such that $\boldsymbol{e} = \mathbf{p}_j = \mathbf{u}_j$, and so $\mathfrak{A}, \mathbf{x} \mapsto \boldsymbol{e} \vDash \neg A(\mathbf{x}) \wedge (\bigvee_i \mathbf{x} = \mathbf{u}_i)$, or $\mathbf{x} \notin [\![\Delta A]\!]_{\mathfrak{A}}$, so $\boldsymbol{e} \in [\![A]\!]_{\mathfrak{A}}$ and for all $1 \leqslant i \leqslant k$, $\boldsymbol{e} \neq \mathbf{p}_i$, so for all $1 \leqslant j \leqslant \beta_A$, $\boldsymbol{e} \neq \mathbf{u}_i$. So $\mathfrak{A}, \mathbf{x} \mapsto \boldsymbol{e} \vDash A(\mathbf{x}) \wedge (\bigwedge_i \mathbf{x} \neq \mathbf{u}_i)$.

The case for $\boldsymbol{e} \notin [\![A^\star]\!]_{\mathfrak{A}}$ is similar. □

Lemma 7.3.3. If ϕ is β-bounded and supported modulo ψ, then $\downarrow\phi$ is first-order definable modulo ψ.

Proof. In the following we assume variable freshness wherever necessary.

For any $\mathfrak{A}, \mu \vDash \psi \wedge \downarrow\phi$, any $A \in \mathbf{Dyn}$, by lemma 7.3.2, we get $\mathfrak{A}, \mathbf{X}^\beta_A \mapsto \mathbf{M}_A \vDash \mathit{Bound}(A)$ for some $\mathbf{M}_A = \mathbf{m}_1, \dots, \mathbf{m}_{\beta_A}$. Consider any set U of arity(A)-sized tuples from $\mathrm{dom}(\mathfrak{A})$ such that $[\![A]\!]_{\mathfrak{A}} \Delta U \subseteq [\![\Delta A]\!]_{\mathfrak{A}}$. For some $0 \leqslant q \leqslant \beta_A$ there are $\mathbf{x}_{k_1}, \dots, \mathbf{x}_{k_q} \in \mathbf{X}^\beta_A$ such that

$$\mathfrak{A}, S' \mapsto U, \mathbf{x}_{k_1} \mapsto \mathbf{m}_{k_1}, \dots, \mathbf{x}_{k_q} \mapsto \mathbf{m}_{k_q} \vDash \mathit{Diff}(S', A, (\mathbf{x}_{k_1}, \dots, \mathbf{x}_{k_q}))$$

Note how the case $q = 0$ makes $\mathit{Diff}(S', A, ())$ degenerate to $\forall \mathbf{x}.\ S'(\mathbf{x}) \leftrightarrow A(\mathbf{x})$.

Moving from the meta language to the logic, we restate the above as:

$$\mathfrak{A}, S' \mapsto U \models \exists \mathbf{X}^{\beta}_{A}.\ \mathit{Bound}(A) \wedge \bigvee_{\mathbf{X} \subseteq \mathbf{X}^{\beta}_{A}} \mathit{Diff}(S', A, \mathbf{X})$$

Since $\psi_1(\mathit{Supp}', \mathbf{Dyn}') \models \Delta A' \subseteq \Delta A$, for any $A' \in \mathbf{Dyn}'$, modulo *Bound* and ψ we have:

$$\begin{aligned}
&\exists \mathbf{X}_A.\ \mathit{Bound}(A) \wedge \forall A'.\ \psi_1(\mathit{Supp}', \mathbf{Dyn}') \rightarrow \psi_2(\mathit{Supp}', \mathbf{Dyn}') \\
\equiv\ &\exists \mathbf{X}_A.\ \mathit{Bound}(A) \wedge \forall A'.\ \psi_1(\mathit{Supp}', \mathbf{Dyn}') \rightarrow \Big(\psi_2(\mathit{Supp}', \mathbf{Dyn}') \wedge \bigvee_{\mathbf{X} \subseteq \mathbf{X}^{\beta}_{A}} \mathit{Diff}(A', A, \mathbf{X})\Big) \\
\equiv\ &\exists \mathbf{X}_A.\ \mathit{Bound}(A) \wedge \forall A'.\ \bigwedge_{\mathbf{X} \subseteq \mathbf{X}^{\beta}_{A}} \Big(\mathit{Diff}(A', A, \mathbf{X}) \wedge \psi_1(\mathit{Supp}', \mathbf{Dyn}')\Big) \rightarrow \psi_2(\mathit{Supp}', \mathbf{Dyn}') \\
\equiv\ &\exists \mathbf{X}_A.\ \mathit{Bound}(A) \wedge \\
&\qquad \forall A'.\ \bigwedge_{\mathbf{X} \subseteq \mathbf{X}^{\beta}_{A}} \Big(\psi_1(\mathit{Supp}', \mathbf{Dyn}') \rightarrow \psi_2(\mathit{Supp}', \mathbf{Dyn}')\Big)\{A'(\mathbf{x}) \mapsto \mathit{Split}(A, \mathbf{X}, \mathbf{x})\}
\end{aligned}$$

Note that the substitution $\{A'(\mathbf{x}) \mapsto \mathit{Split}(A, \mathbf{X}, \mathbf{x})\}$ captures the arguments of A' and uses them as arguments for *Split*; the substitution is simply applying the equivalence $\mathit{Diff}(A', A, \mathbf{X}) = \forall \mathbf{x}.\ A'(\mathbf{x}) \leftrightarrow \mathit{Split}(A, \mathbf{X}, \mathbf{x})$ at the syntax level.

Since all occurrences of A' have been removed, the $\forall A'.$ can be dropped. Starting from the formula provided by lemma 7.3.1, repeated application of the above for all $A \in \mathbf{Dyn}$ removes all 2nd-order quantification from $\downarrow\phi$: As syntactic sugar, define

$$\psi := \psi_1(\mathit{Supp}', \mathbf{Dyn}') \rightarrow \psi_2(\mathit{Supp}', \mathbf{Dyn}')$$

With a bounded support formula ϕ, and $\mathbf{Dyn} = (A_1, \ldots, A_p)$, for any $1 \leqslant i \leqslant p$, any tuple $\mathbf{Y}$ of arity(A_i)-sized tuples of variables, define the substitution

$$\sigma_i(\mathbf{Y}) := A'_i(\mathbf{x}) \mapsto \mathit{Split}(A_i, \mathbf{Y}, \mathbf{x})$$

Modulo ψ, we have:

$$\begin{aligned}
\downarrow\phi \equiv\ &\downarrow\phi \wedge \bigwedge_{A_i \in \mathbf{Dyn}} \exists \mathbf{X}^{\beta}_{A_i}.\ \mathit{Bound}(A_i) \\
\equiv\ &\exists \mathbf{X}^{\beta}_{A_1}, \ldots, \mathbf{X}^{\beta}_{A_m}.\ \downarrow\phi \wedge \mathit{Bounds} \qquad\qquad \text{(where } \mathit{Bounds} := \textstyle\bigwedge_{A_i \in \mathbf{Dyn}} \mathit{Bound}(A_i)\text{)} \\
\equiv\ &\exists \mathbf{X}^{\beta}_{A_1}, \ldots, \mathbf{X}^{\beta}_{A_m}.\ x\phi \wedge (\forall A'_0, \ldots, A'_m.\ \psi) \wedge \mathit{Bounds} \\
\equiv\ &\exists \mathbf{X}^{\beta}_{A_1}, \ldots, \mathbf{X}^{\beta}_{A_m}.\ \phi \wedge \left(\forall A'_0, \ldots, A'_{n-1}.\ \bigwedge_{\mathbf{Y}_{A_m} \subseteq \mathbf{X}^{\beta}_{A_m}} \psi\{\sigma_m(\mathbf{Y}_{A_m})\}\right) \wedge \mathit{Bounds} \\
&\vdots \\
\equiv\ &\exists \mathbf{X}^{\beta}_{A_1}, \ldots, \mathbf{X}^{\beta}_{A_m}.\ \phi \wedge \left(\bigwedge_{\mathbf{Y}_{A_1} \subseteq \mathbf{X}^{\beta}_{A_1}} \cdots \bigwedge_{\mathbf{Y}_{A_m} \subseteq \mathbf{X}^{\beta}_{A_m}} \psi\{\sigma_m(\mathbf{Y}_{A_m})\} \ldots \{\sigma_1(\mathbf{Y}_{A_1})\}\right) \wedge \mathit{Bounds}
\end{aligned}$$

□

7.3.3. Simplifying FO quantifiers

We can show the level of quantifier nesting is not increased much by minimisation.

Lemma 7.3.4. If ϕ is supported, ϕ is bounded modulo ψ, η is a first-order quantifier alternation prefix, and $\phi \in \eta$ and $\neg\phi \in \eta$, then $\downarrow \phi$ is equivalent modulo ψ to an FO formula with a quantifier prefix in $\exists^*\eta\forall^*$.

Proof. By inspection of the first-order formula equivalent to $\downarrow\phi$ (modulo ψ) given by lemma 7.3.3, and by trivial syntactic manipulation: $\bigwedge_{\mathsf{Y}_{A_i} \subseteq \mathsf{X}_{A_i}} (\psi_1(\mathit{Supp}', \mathbf{Dyn}') \to \psi_2(\mathit{Supp}', \mathbf{Dyn}'))\{A_i'(\mathbf{x}) \mapsto \mathit{Split}(A_i, \mathbf{Y}_{A_i}, \mathbf{x})\}$ is a conjunction of formulas with quantifier prefix in η (by assumption on ϕ), and $\mathit{Bound}(A_i)$ contains only universal quantifiers in prenex form. □

Definition. Let $d \geqslant 0$. In an n-BFL, the formula $\mathit{SameTree}(x, y)$ specifies that x and y are in the same tree. The formula $\delta_{\leqslant d}(x, y)$ specifies that x and y are at distance $\leqslant d$. We skip the proofs for each.

$$\mathsf{SameTree}(x, y) := \mathit{parent}^n(x) = \mathit{parent}^n(y)$$

$$\delta_{\leqslant d}(x_0, y_d) := \exists\,\{y_{d'-1}, x_{d'} \mid 0 < d' \leqslant d\}.\ \left(\bigwedge_{0 \leqslant i \leqslant d} \mathsf{SameTree}(x_i, y_i)\right) \wedge \left(\bigwedge_{0 \leqslant i < d} y_i = x_{i+1} \vee \mathsf{Link}(y_i, x_{i+1}) \vee \mathsf{Link}^\star(y_i, x_{i+1})\right)$$

■

The goal of the following lemma is to remove an existential quantifier in front of a formula. Recall that $\ell_{\Theta_n}(d)$ bounds the size of any d-sized ball in any $\mathfrak{A} : \Theta_n$ (cf. lemma 4.3.6 p.67).

Lemma 7.3.5. If $d \geqslant 0$, and ψ is a formula then $\exists x.\ \delta_{\leqslant d}(x, y) \wedge \psi$ is equivalent modulo $\mathcal{T}_n$ to:

$$\bigvee_{1 \leqslant k \leqslant \ell_{\Theta_n}(d)} \left(\left(\exists x_1, \ldots, x_k.\ \bigwedge_{1 \leqslant i \leqslant k} \delta_{\leqslant d}(x_i, y) \wedge \bigwedge_{1 \leqslant j \neq i \leqslant k} x_i \neq x_j\right) \wedge \forall x_1, \ldots, x_k.\ \left(\bigwedge_{1 \leqslant i \leqslant k} \delta_{\leqslant d}(x_i, y) \wedge \bigwedge_{1 \leqslant j \neq i \leqslant k} x_i \neq x_j\right) \to \bigvee_{1 \leqslant i \leqslant k} \psi\{x \mapsto x_i\}\right)$$

Proof. By lemma 4.3.6, there are at most $\ell_{\Theta_n}(d)$ nodes at distance at most d from y.

Suppose $\mathfrak{A}, \mu \vDash \psi$ and $\delta_{\mathfrak{A}}(\mu(x), \mu(y)) \leqslant d$. Let $1 \leqslant k \leqslant \ell_{\Theta_n}(d)$ be the number of elements at distance at most d from $\mu(y)$. The kth disjunct of the above sentence is satisfied by taking those k elements as existential witnesses of for $x_1, \ldots, x_k$ (left conjunct): they are at distance $\leqslant d$ from y, all different. The only universal witnesses for $x_1, \ldots, x_k$ (right conjunct) that satisfy the premise are those k elements, and one of them is $\mu(x)$.

The other way around, let θ_k be the satisfied disjunct; by the existential part of θ_k, its universal part is not vacuously true. □

Each disjunct in the formula above supposes a different number of distinct elements in $\mathcal{B}_{\mathfrak{A}}(\mu(y), d)$. For each possible answer, the universal part affirms that at least of those elements can play the role of x in ψ. We could reduce the number of disjunctions by counting the number of trees around an element instead of the number of elements; then the $x_1, \dots, x_k$ would be tree roots and the final disjunct $(\psi\{x \mapsto x_i\})$ would be over all descendants of the correct root.

Theorem 7.3.6. If φ is $(\mathcal{V}, d)$-preserved then $\varphi \wedge \mathit{Support}_D$ is bounded modulo $\mathcal{T}_n$.

Proof. Let $\mathfrak{A} : \Theta_n$ such that $\mathfrak{A}, \mu \models \downarrow (\varphi \wedge \mathit{Support}_D)$. Let $A \in \mathbf{Dyn}$, $\boldsymbol{e} \in [\![\Delta A]\!]_{\mathfrak{A}}$ and $a \in \boldsymbol{e}$. By lemma 4.3.5, if $a \notin \mathcal{B}_{\mathfrak{A}}(\mu(\mathcal{V}), d+1)$ then there is a $(\mu(\mathcal{V}), d)$-sub $\mathfrak{B}$ of $\mathfrak{A}$ such that $\mathfrak{B} \neq \mathfrak{A}$ and $\mathfrak{B} \models \mathit{Support}_D$. So $\mathfrak{B} <_\alpha \mathfrak{A}$, by definition of subs. By lemma 4.3.3, $\mathfrak{B} : \Theta_n$. And since φ is $(\mathcal{V}, d)$-preserved, $\mathfrak{B}, \mu \models \varphi$. Absurd by minimality of $\mathfrak{A}$. So $[\![\Delta A]\!]_{\mathfrak{A}} \subseteq \mathcal{B}_{\mathfrak{A}}(\mu(\mathcal{V}), d+1)^{\mathrm{arity}(A)}$.

By lemma 4.3.6, $|\mathcal{B}_{\mathfrak{A}}(\mu(\mathcal{V}), d+1)| \leqslant |\mathcal{V}|\, \ell_{\Theta_n}(d+1)$. □

Lemma 7.3.7. If φ' is $(\mathcal{V}, d)$-preserved, in $\exists^*\forall^*$ and in $\forall^*\exists^*$ then $\downarrow(\varphi' \wedge \mathit{Support}_D)$ is in $\exists^*\forall^*$ and in $\forall^*\exists^*$ modulo $\mathcal{T}_n$.

Proof. Let $\varphi \coloneqq \varphi' \wedge \mathit{Support}_D$. For $\downarrow\varphi \in \exists^*\forall^*$ modulo $\mathcal{T}_n$, we get β-boundedness modulo $\mathcal{T}_n$ (for some $\beta : \mathbf{Dyn} \to \mathbb{N}$) with lemma 7.3.6, φ is supported by definition, and so we get the syntactic fragment with lemma 7.3.4.

Inspecting the formula equivalent to $\downarrow\varphi$ modulo $\mathcal{T}_n$ as shown in 7.3.3, we have:

$$\downarrow\varphi \equiv \exists \mathbf{X}_{A_1}, \dots, \mathbf{X}_{A_m}.\ F$$

where

$$F \coloneqq \varphi \wedge \left(\bigwedge_{\mathbf{Y}_{A_1} \subseteq \mathbf{X}_{A_1}} \cdots \bigwedge_{\mathbf{Y}_{A_m} \subseteq \mathbf{X}_{A_m}} \psi\{\sigma_m(\mathbf{Y}_{A_m})\} \dots \{\sigma_1(\mathbf{Y}_{A_1})\} \right) \wedge \mathit{Bounds}$$

It is easy to syntactically check that $F \in \forall^*\exists^*$ (remember that $\varphi' \in \exists^*\forall^*$, $\varphi' \in \forall^*\exists^*$, and that $\mathit{Support}_D$ is a universal sentence). Now, we show that each witness in $\exists \mathbf{X}_{A_i}$ can be always be found at distance $\leqslant d+1$ from $\mu(\mathcal{V})$ for any $\mathfrak{A} : \Theta_n$ such that $\mathfrak{A}, \mu \models \downarrow\varphi$.

As a reminder, $\mathit{Bound}(A_i) \coloneqq (\forall x.\ A_i^\star(x) \leftrightarrow A_i(x)) \vee \mathit{Diff}(A^\star, A_i, \mathbf{X}_{A_i})$ and is satisfied for every $A_i \in \mathbf{Dyn}$. Let $\mathfrak{A}, (\mathbf{X}_{A_i} \mapsto \mathbf{M}_{A_i})_{1 \leqslant i \leqslant m}, \mu \models F$. If $\mathit{Bound}(A_i)$ is satisfied by its 1st disjunct, then $[\![\Delta A_i]\!]_{\mathfrak{A}} = \emptyset$ and the choice of witness is irrelevant.

Otherwise, by Diff, the interpretations for $\mathbf{X}_{A_i}$ are exactly the members of $[\![\Delta A_i]\!]_{\mathfrak{A}}$. By lemma 4.3.5, for every $\boldsymbol{e} \in [\![\Delta A]\!]_{\mathfrak{A}}$ and every $a \in \boldsymbol{e}$, there is some $c \in \mu(\mathcal{V})$ such that $\delta_{\mathfrak{A}}(a, c) \leqslant d+1$.

By the above, we have modulo $\mathcal{T}_n$:

$$\downarrow\varphi \equiv \exists \mathbf{X}_{A_1}, \dots, \mathbf{X}_{A_m}.\ \Big(\bigwedge_{1 \leqslant i \leqslant m} \bigwedge_{x \in \mathbf{X}_{A_i}} \bigwedge_{x \in x} \bigvee_{y \in \mathcal{V}} \delta_{\leqslant d+1}(x, y) \Big) \wedge F$$

Repeated application of lemma 7.3.5 yields that modulo $\mathcal{T}_n$, $\downarrow\varphi$ is equivalent to a disjunction of formulas with the following shape:

$$(\exists \mathbf{X}_{A_1}, \dots, \mathbf{X}_{A_m}.\ G_1) \wedge \forall \mathbf{X}_{A_1}, \dots, \mathbf{X}_{A_m}.\ G_1 \to F'$$

where G_1 is existential and $F' \in \forall^*\exists^*$. G_1 is a conjunction of existential formulas of the form $\delta_{\leqslant d+1}(x, y)$ or $x \neq y$. F' is a disjunction of formulas in $\forall^*\exists^*$ (see the details of the construction in lemma 7.3.5), so $\downarrow\varphi \in \forall^*\exists^*$ modulo $\mathcal{T}_n$.

□

We can move to the main result of this section:

7.3.4. Main theorem for elimination of $\downarrow$

Theorem 7.3.8. If $\mathcal{V}\,; d \vDash_{\triangleleft} \phi$, then $\phi \wedge \mathcal{T}_n$ is in both $\exists^*\forall^*$ and $\forall^*\exists^*$.

Proof. Remember that $\mathcal{T}_n$ is a universal sentence, so it suffices to prove that ϕ is in both $\exists^*\forall^*$ and $\forall^*\exists^*$ modulo $\mathcal{T}_n$. We proceed by induction on the derivation, proving for all $\mathcal{V}, d$.

DYNAMIC$_\triangleleft$, STATIC$_\triangleleft$, INVARIANT$_\triangleleft$, WEAK$_\triangleleft$, BOOL$_\triangleleft$,GUARD$_\triangleleft$ Trivial.

$\exists$-GUARD$_\triangleleft$ $\phi = \exists x.\ \alpha(x,y) \wedge \phi' \in \exists^*\forall^*$ by induction hypothesis on ϕ'. Moreover, $x \neq y$, so by functionality of Link, Link^* (given by $\mathcal{T}_n$) and equality, we have $\phi \equiv (\exists x.\ \alpha(x,y)) \wedge (\forall x.\ \alpha(x,y) \rightarrow \phi)$; since $\phi' \in \forall^*\exists^*$ by induction hypothesis, $\phi \in \forall^*\exists^*$ as well.

$\forall_\triangleleft$ $\forall x.\ \phi \in \forall^*\exists^*$ using the induction. For $\exists^*\forall^*$: no subformula of ϕ is ψ such that $x \in \langle\psi\rangle$ and of the form $\psi =\downarrow\psi'$ since CIRCUM$_\triangleleft$ adds all variables to $\mathcal{V}$ and adds 1 to d, and INVARIANT$_\triangleleft$ (which removes variables from $\mathcal{V}$) requires a formula with $d = 0$. So $\phi \in$ FO. We now proceed by induction over the number of $\exists$ quantifiers under $\forall x.$.

We put ϕ in conjunctive normal form, treating quantified $\exists$ subformulas as atoms, and distributing $\forall x.$ over $\wedge$, pushing it inside other toplevel $\forall$, and pushing universal quantifiers up. Note that by distributing $\forall x.$ over $\wedge$, the number of $\exists$ under $\forall x.$ does not increase.

Every conjunct of the resulting formula is of the form

$$\forall x.\ \bigvee \theta_i$$

where each θ_i is a literal or a formula of the form

$$\exists y.\ \alpha(y,z) \wedge \theta_i'$$

where $\mathcal{V}', d' \vDash_{\triangleleft} \theta_i'$ for some $\mathcal{V}', d'$, since the existential subformulas have been treated as atoms, pre formulas are all provable (for INVARIANT$_\triangleleft$) and guards are all provable (for GUARD$_\triangleleft$). Moreover $x \neq z$: by assumption $x \notin \mathcal{V}$ (otherwise $\forall_\triangleleft$ could not have been applied), and $\exists$-GUARD$_\triangleleft$ adds z to the context. The only rule that could have removed z is INVARIANT$_\triangleleft$, but that requires $d = 0$, and $\exists$-GUARD$_\triangleleft$ increases d by 1 (rules are monotone w.r.t. d). Now, by induction on the number of θ_i' where x occurs. If there is none, every $\exists$ and $\forall x.$ can all be swapped and we are done.

Otherwise the clause is of the form

$$\forall x.\ (\exists y.\ \alpha(y,z) \wedge \theta_i') \vee \theta$$

where x occurs in θ_i'. By the same argument as in the previous case ($\exists$-GUARD$_\triangleleft$), this is equivalent modulo $\mathcal{T}_n$ to:

$$\begin{aligned} &(\forall x.\ \theta \vee \exists y.\ \alpha(y,z)) \wedge (\forall x.\ \theta \vee \forall y.\ \alpha(y,z) \rightarrow \theta_2) \\ \equiv &(\forall x.\ \theta \vee \exists y.\ \alpha(y,z)) \wedge \forall y.\ (\forall x.\ \theta \vee \neg\alpha(y,z) \vee \theta_2) \qquad \text{(up to variable renaming)} \end{aligned}$$

In the left conjunct, there is one less θ_i' where x occurs, so done by the innermost induction hypothesis. In the right conjunct, there is one less $\exists$ under $\forall x.$ and $\theta \vee \alpha(y,z) \vee \theta_2$ is $\vDash_{\triangleleft}$-provable, so done by the outermost induction hypothesis.

Circum$_\lhd$ Lemma 4.3.3 implies that *Symlink* is $(\emptyset, 0)$-preserved, so by the premise of Circum$_\lhd$ and theorem 4.4.5, $\phi \wedge$ *Symlink* is $(\mathcal{V}, d)$-preserved. Moreover, *Symlink* is a universal sentence so $\phi \wedge$ *Symlink* is in $\exists^*\forall^*$ and $\forall^*\exists^*$. So by lemma 7.3.7 (applicable thanks to the induction hypothesis), and Weak$_\lhd$, we are done.

$\square$

8. Conclusion

8.1. Related work

Circumscription The use of circumscription for nonmonotonic reasoning dates back to McCarthy [McCarthy, 1980], and attempts at inferring non-changes logically can be found by following mentions of the frame problem [Reiter, 1991]. We use an instanciation of unified circumscription, a new flavor of circumscription which generalises [Doherty et al., 1998]. Previous works on taming circumscription require global syntactic properties of the formulas [Doherty et al., 1997, Nonnengart et al., 1999, Conradie, 2006] and only consider satisfiability or FO-definability of circumscribed formulas; we introduce a modular method for building circumscribed formulas that live in a tight fragment of FO. The resulting classes of formulas are incomparable with those presented here. Moreover, the goal of circumscription is usually to approximate commonsense reasoning. In contrast, we want to use nonmonotonicity for the purpose of graph rewriting rule synthesis.

There is a wide variety of forms of circumscription, and some make the connection to contemporary logic programming by representing semantics such as stable model and FLP [Ferraris et al., 2007, Ferraris et al., 2011, Wan et al., 2014, Zhang and Ying, 2010, Yang et al., 2011, Bartholomew et al., 2011]; we hope other connections can be made between the form of circumscription we developed here and these fields.

Biological modelling Logical approaches to biological and chemical modelling exist in many flavors, using linear logic or other resource-aware logics [Despeyroux, 2016, Boniolo et al., 2010], or logic rules for specification and modality for queries [Eker et al., 2002], or explicitly representing reactions as first-class citizens and using circumscription for describing weakest preconditions to reactions [Doherty et al., 2004]. The focus, so far, has been on verification of models or pure knowledge representation. Here, we offer a classically model-based approach and maintain intuitive semantics. We follow the tradition of taking graph rewriting as a domain-specific language for biology [Boutillier et al., 2018, Chylek et al., 2014, Faeder et al., 2009] and we try to retain enough expressivity for large-scale, natural knowledge representation while ensuring reasonable tractability. More generally, logical approaches such a boolean networks are widely used to model gene interactions [Novère, 2015].

Another approach is [Basso-Blandin et al., 2015], which extends Kappa using slice categories over a particular kind of graph (reminiscent of bigraphs) and their homorphisms. The slice is over a master graph which represents all possible entities and their interactions. "Nuggets" of knowledge are given as graphs together with a homomorphism towards the master. Nuggets of knowledge are updated using categorical operations, and in some cases a set of knowledge nuggets can be compiled to Kappa. This approach seems promising for the connection between ontologies in biology [Stevens et al., 2000, Bard, 2004] and rule-based execution.

Transition structures We compare with two well-known approaches. Hoare logic is often about verifying properties of programs, by obtaining a proof of $\{P\}p\{Q\}$ for statements P, Q and a program p. Corresponding assertions in our setting would be of the form $P^\circ \to Q^\star$, with a question mark as a placeholder for p: we are concerned with synthesising p from a specification. The p we wish to synthesize, however, is not any program satisfying $P^\circ \to Q^\star$; we want it to be among the "best" programs, in the sense that it also edits the memory as little as possible. Other differences are that our programs run in constant time, and that we want to mix pre- and post-conditions more freely. There have been approaches to Hoare logic-based program synthesis ([Mamouras, 2016]); in the future work section we describe an idea for "rule merging" which would yield something similar to a Hoare sequence rule $\{P\}p\{Q\}, \{Q\}p'\{R\} \Rightarrow \{P\}p;p'\{R\}$.

A logic on pre- and post-conditions such as the one introduced here naturally makes one think of an extension to more frames, either by adding new copies of **Dyn**, such as $\mathbf{Dyn}^{\star\star\cdots}$, or in the style of [Reiter, 2001] by adding natural numbers and an additional dimension to every dynamic statement: one can then write for instance $\mathsf{A}(7, x)$ instead of $\mathsf{A}(x)$ to mean that $\mathsf{A}(x)$ is true at frame 7, or even $\forall i : \text{int}.\ \mathsf{A}(i, x) \to \mathsf{A}(i+1, x)$. A modal version of this approach would be dynamic logic, but 1) we see this approach as a logic on *traces* rather on rules, about which see [Laurent et al., 2018] for a trace query language on Kappa traces, and 2) the crucial notion of minimality, especially on an unbounded number of frames, could resurrect variants of the frame problem [Reiter, 1991].

Guarded fragment Many constructions in this thesis are based on *guards*. After the introduction of $\vDash_\triangleleft$, they become quite constrained (they are morally of bounded degree) anyway, but before that, one could try to extend the correctness results on $\vDash_p$ to looser guards, such as the loosely guarded fragment [Hodkinson, 2002], or even the guarded negation fragment [Bárány et al., 2011].

8.2. Conclusion

This thesis is an attempt to lay the foundations for a more capable molecular biology modelling framework. It takes rule-based modelling as a starting point and extends it through logic. Models are transitions between states. Transition signatures are separated into precondition ($\mathsf{Link}, \mathsf{Has}, \mathsf{A}, \ldots$), postcondition ($\mathsf{Link}^\star, \mathsf{Has}^\star, \mathsf{A}^\star, \ldots$) and static information (N, f, parent). For instance, $\mathsf{Enzyme}(x) \wedge \neg\mathsf{Active}(x) \wedge \mathsf{Active}^\star(x)$ means that x is an enzyme and becomes active. Any context is allowed. Anything else can happen. State signature drop the postcondition part.

A reasonable specialisation is the class of n-BFLs, where the static part describes height-bounded trees and where $\mathsf{Link}, \mathsf{Link}^\star$ are functional. The static part represents the inner protein structure, while the dynamic part represents transient protein state and protein-protein interactions.

Even modulo the theory of n-BFLs, first-order satisfiability is not decidable. But decidability in the $\exists^*\forall^*$ fragment is decidable.

There is a change order $\trianglelefteq$ between transitions. If $\mathfrak{A} \trianglelefteq \mathfrak{B}$, $\mathfrak{A}$ and $\mathfrak{B}$ start from the same state but $\mathfrak{A}$ "does less" than $\mathfrak{B}$. This order is relevant to biological modelling because execution of protein-protein interactions requires a closed-world policy regarding which changes can occur. An operator $\downarrow$ is added to first-order logic. For any formula ϕ, $\downarrow\phi$ is the set of $\trianglelefteq$-minimal models of ϕ. $\downarrow$ represents the end of the data gathering phase and the beginning of the execution phase.

Circumscription is a class of 2nd-order macros created by [McCarthy, 1980], used to capture minimal models of formulas along various orders. Unified circumscription generalises various common forms of circumscription, and can be instantiated to define $\downarrow$. So it is possible to define $\downarrow$ syntactically using 2nd-order logic.

Transitions can be summarised by transition patterns, of the form $(\mathfrak{A}, K)$ with $K \subseteq \mathsf{dom}(\mathfrak{A})$. While an n-BFL $\mathfrak{A}$ can be "found" in another n-BFL $\mathfrak{B}$ if there is an embedding from $\mathfrak{A}$ to $\mathfrak{B}$, a pattern adds the constraint that the embedding must add no new Link (or $\mathsf{Link}^\star$) to elements of K. Without patterns, negative constraints such as $\forall y.\ \neg\mathsf{Link}(c, y)$ could not be encoded. Embeddings that agree with that constraint are matches.

There are two ways to generate (finite or infinite) sequences of sates. One is operational, based on sets of rules (a rule is just a pattern with more information): given a partial match from a rule to a state, a next state is constructed. The other is denotational, based on sets of transitions: any sequence from the set is correct if the post- and preconditions of every successive transition are equal. All sequences are defined relative to some starting state.

The main contribution of this thesis is to provide conditions where those two notions coincide and where the operational part can be finitely described:

For every set of transitions, there is an associated set of canonical patterns (and corresponding rules). If a formula ϕ is provable in the deductive system $\vdash_\rho$ then, for some starting states, the denotational runs of $[\![\phi]\!]$ are exactly the executions induced by the canonical rules associated to $[\![\phi]\!]$.

A rule-based rewriting engine starts from a file, which is a finite object. In general, the canonical rules associated to $[\![\phi]\!]$ may be an infinite set. If a formula ϕ is provable in another deductive system, $\vdash_\triangleleft$, then that set of rules is finite. The execution of those rules may not match the denotational runs; for that, one needs $\vdash_\rho$.

This is the desired correspondence between a denotational notion of runs and an operation notion of execution, given by a finite object. That finite object is essentially a subset of the models of a formula ϕ within a bounded size, so it can effectively be constructed.

An additional property of $\vdash_\triangleleft$ is that, modulo the theory of n-BFLs, the formulas it proves are all in the $\exists^*\forall^*$ fragment and also in the $\forall^*\exists^*$ fragment. But they are not negation-closed. Given the earlier decidability result, this opens the door to querying. The elimination of $\downarrow$ happens thanks to its 2nd-order, syntactic definition.

A special-purpose syntax for programs is here to make biological modelling easier. Every program $\mathtt{P}$ corresponds to a formula $\phi_\mathtt{P}$, and $\phi_\mathtt{P}$ is always provable both in $\vdash_\triangleleft$ and $\vdash_\rho$, up to equivalence.

8.3. Future work

Application to a biological use case In the short term, we plan on applying the *program* syntax to a relevant biological use case. However, we believe that most of the practical usefulness of the logic developed here will after the development of a higher-level language that is more comfortable to use and compiles to the iota logic.

Complexity of the fragments There are lower bounds to establish on satisfiability in $\exists^*\forall^*$ modulo $\mathcal{T}_n$ as well as on $\vdash_\rho$ and $\vdash_\triangleleft$-provable formulas.

Stochastic rates To accurately represent biological observations, attaching reaction rates to each rule is mandatory. Given a state and a set of rules, every "match" (in the sense of some graph morphism) from a rule to the state is counted. The probability of rewriting with one match is weighted by the rate attached to the rule which is source of the match.

Rewriting in our setting is nondeterministic. We intend to introduce rates at the level of programs: a program would be of the form

$$\theta_1 \; \mathtt{at\,rate}_1 \; ; \; \ldots \; ; \; \theta_k \; \mathtt{at\,rate}_k \; (\ldots)$$

where each $\mathtt{rate}_i$ is e.g. a floating-point number. The main difficulty is finding a representation of the canonical patterns such that the number of transitions at the denotational level is equal to the number of match at the pattern level.

Better $\vDash_{\triangleleft}$ Currently, a formula such as

$$\mathit{Symlink} = \forall xy.\ (\mathsf{Link}(x,y) \to \mathsf{Link}(y,x)) \wedge (\mathsf{Link}^\star(x,y) \to \mathsf{Link}^\star(y,x))$$

is not provable. In general, if there is a guard $\alpha(x,y) \to \ldots$, the "$\ldots$" may only contain dynamic information about x or y, but not both. We think the requirement can be loosened: binary dynamic atoms that mention *both* x and y can be allowed (but dynamic atoms that only mention x cannot be allowed together with dynamic atoms that only mention y). This should not require changing the definition of preservation; we think it should be possible to define a class of *nice* clauses (as we did with good and great clauses in chapter 3), and add a rule of the form

$$\mathcal{C}\text{ nice}\quad \frac{}{\emptyset, 0 \vDash_{\triangleleft} \mathcal{C}^\circ \wedge \mathcal{C}^\star}$$

The rule would replace both $\textsc{Invariant}_{\triangleleft}$ and $\textsc{Guard}_{\triangleleft}$. $\textsc{Invariant}_{\triangleleft}$ would be admissible, using a technique similar to the proof of lemma 5.2.2, and exactly the same formulas would be provable, albeit as tightly (context-wise) as with $\textsc{Guard}_{\triangleleft}$. It seems that the following definition of a *nice clause* $\mathcal{C}$ should work: there is some x mentioned in every dynamic literal, and every $y \neq x$ mentioned in a dynamic literal is mentioned in a negative dynamic literal.

Rule merging Consider F_0 and F_1 two formulas on Θ. Let $\mathrm{Shift}(F_1)$ be F_1 where every starred symbol in F_1 is replaced with a *doubly starred symbol*, and every unstarred symbol is replaced with a starred symbol. For instance, $A^\star$ becomes $A^{\star\star}$ and A becomes $A^\star$. We would be using a signature $\Theta' = (\mathbf{Stat}, \mathbf{Dyn}, \mathbf{Dyn}^\star, \mathbf{Dyn}^{\star\star})$ with static symbols, normal dynamic symbols, starred symbols and doubly starred symbols. The formula $F_0 \wedge \mathrm{Shift}(F_1)$ would be a Θ'-formula.

We could of course extend this to k-state transitions, which opens questions about minimisation (but introduces issues related to the frame problem [Reiter, 1991] that we have mostly avoided in this thesis). But the 3-place case is particular since we don't need more to ask this question:

Is there a Θ-formula F such that $[\![F]\!]$ is the projection of $\mathbf{Stat}, \mathbf{Dyn}, \mathbf{Dyn}^{\star\star}$ from the models of $[\![F_0 \wedge \mathrm{Shift}(F_1)]\!]$?

If yes, we could iteratively apply this process and have formulas that characterize the n-step accessible states from any starting state. More prosaically, it means we could construct new rules by combining existing rules in a new dimension (by gluing them end-to-end with the shift-conjunction method given above for F_0 and F_1).

Suppose that, for every interpretation of $\mathbf{Stat}, \mathbf{Dyn}, \mathbf{Dyn}^{\star\star}$, $[\![F_0 \wedge \mathrm{Shift}(F_1)]\!]$ only allows a single middle state (i.e. the semantics of $\mathbf{Dyn}^\star$ are uniquely defined by the semantics of $\mathbf{Stat}, \mathbf{Dyn}, \mathbf{Dyn}^{\star\star}$). In that case Beth's definability theorem answers in the positive: there is a Θ-formula as we wish. We wonder if there are extensions of Beth's theorem that could hold when the middle state is not uniquely defined but may only be one among a finite number of states, with that bound a function of F_0 and F_1. Under $\downarrow$ this constraint is realistic.

Unguarded $\exists$ in $\vDash_{\triangleleft}$ As mentioned page 69, it should be possible to allow unguarded existential quantification in $\vDash_{\triangleleft}$ by changing the definition of preservation. Instead of drawing boundaries around the image of an interpretation, the definition would require the existence of a bounded-size protected set. We avoided this approach because it does not immediately yield *where* these protected elements are, and the proofs from chapter 4 on eliminating $\downarrow$ from $\vDash_{\triangleleft}$-provable formulas relies on knowing the location of these protected elements.

Removing Support Even though support is used heavily throughout, we think it may be possible to not rely on it so much and simplify proofs in the process. First, state could be on the signature $\Theta^{\circ} \setminus \{\mathsf{Has}\}$, and the state joining operation $\langle \cdot, \cdot \rangle$ could add populate the interpretations of Has and $\mathsf{Has}^{\star}$ depending on whether an element is in the domain of the left state and/or the right state. This way, all transitions would be immediately decomposable, and would also be immediately strongly supported as well as dynamically supported.

The harder part is that support seems to be necessary in the proof of $\downarrow$-elimination when the domain quantification is removed, but modulo boundedness, it may be possible to eliminate domain quantification by quantifying on a finite number of "possibly removed" elements (recall that in circumscription domain quantification only needs to consider subdomains of the "current" one, so if there are a finite number of candidates for removal, there are a finite number of subdomains to consider).

Effective rule synthesis We have so far avoided the topic of complexity for rule synthesis. For a formula ϕ, can we write down the rules induced by $\mathcal{P}(\phi, \|\phi\|)$ in reasonable time? Those patterns are of size $\mathcal{O}(|\mathbf{Stat}_{\mathrm{fun}}|^{2\|\phi\|n^2})$ (see lemma 4.3.6 page 67), i.e. exponential in the quantifier depth of ϕ and in n^2. For a given $s \geqslant 0$, the number patterns of size s (up to iso) is bounded by $2^{su} \times (s+1)^{2s}$, where u is the number of unary predicates in Θ. The first term is for the names and dynamic properties of elements. The second term is for each functional link relations (Link and $\mathsf{Link}^{\star}$). So we are a priori in double exponential time.

First, an observation: formulas induced by programs are of the form $\bigvee \downarrow\psi_i$, and each pattern set can be synthesised in parallel. It seems reasonable to hope for a large number of small ψ_is.

Second, we have good hope of finding conditions where there exists a pattern set P such that $\mathrm{Insts}(P) = \mathrm{Insts}(\mathcal{P}(\phi, \|\phi\|))$ (same instances implies same runs), and such that each $p \in P$ is of size *polynomial* in the number of quantifiers of ϕ: quantifier depth bounds the maximum *distance* between "important" pattern elements, but the number of quantifiers bounds the *number* of "important" pattern trees. Trees have size exponential in n, however.

The size of P itself, however, would be exponential in the number of quantifiers of ϕ and in n.

Full-blown specification language First, an extension of programs $\mathtt{P}$: we would like to extend the syntax with

$$\texttt{react} \ +J_1 \to J_1', \ldots, +J_m \to J_m'$$

where each J_i has the same form as the I_j's in the definition of $\mathtt{P}$, J_i' is quantifier-free, $\langle J_i' \rangle \subseteq \langle J_i \rangle$ and $+J_i \to J_i'$ is compiled to

$$\forall \neg J_i \to J_i^{\star} \to J_i'$$

There may be additional constraint on the polarity of literals and whether J_i' is purely post. In plain english, this would add universal invariants of the form "if J_i becomes true, ensure J_i'". The idea is that with formula of the form $+A \to B$, the variables mentioned in B are already known to be active. So if their tree is cleared, $+A$ becomes false, validating the implication. We are confident that this type of formula is provable in $\vdash_\rho$. $\vdash_\triangleleft$ may need to be extended.

Second, a longer-term goal is to develop a full-fledged programming language for molecular biologists. Any valid program would compile to a formula provable both in $\vdash_\rho$ and $\vdash_\triangleleft$.

A. Additional proofs

To make reading easier, we reproduce the definition of $\mathcal{T}_n$ given page 61:

$$\mathcal{T}_n := \mathit{Symlink} \wedge \mathit{Funlink} \wedge \mathcal{T}'_n$$

where

$$\begin{aligned}
\mathcal{T}'_n &:= \forall x. \mathit{PSpec}(x) \wedge H^n(x) \\
\mathit{PSpec}(x) &:= \Big(\bigwedge_{f \in \mathbf{Stat}_{\mathrm{fun}}} f(x) \neq x \to \mathsf{parent}(f(x)) = x \Big) \\
&\quad \wedge \Big(\mathsf{parent}(x) \neq x \to \bigvee_{f \in \mathbf{Stat}_{\mathrm{fun}}} f(\mathsf{parent}(x)) = x \Big) \\
H^i(x) &:= \bigwedge_{f \in \mathbf{Stat}_{\mathrm{fun}}} f(x) \neq x \to H^{i-1}(f(x)) && \text{(for } 0 \leqslant i < n\text{)} \\
H^{-1}(x) &:= \bot \\
\mathit{FunLink}' &:= \forall x, y, z.\ \mathsf{Link}(x, y) \to \mathsf{Link}(x, z) \to y = z \qquad && \mathit{FunLink} := \mathit{FunLink} \wedge \mathit{FunLink}^\star \\
\mathit{Symlink}' &:= \forall x, y.\ \mathsf{Link}(x, y) \to \mathsf{Link}(y, x) && \mathit{Symlink} := \mathit{Symlink} \wedge \mathit{Symlink}^\star
\end{aligned}$$

Lemma 4.1.1. If $\mathfrak{A} : \Theta$, $\mathfrak{A} : \Theta_n$ iff $\mathfrak{A} \models \mathcal{T}_n$.

Proof. Trivially, $\mathfrak{A} \models \mathit{Symlink} \wedge \mathit{FunLink}$ iff $[\![\mathsf{Link}]\!]_{\mathfrak{A}}$ and $[\![\mathsf{Link}^\star]\!]_{\mathfrak{A}}$ are symmetric and functional. Let $p := [\![\mathsf{parent}]\!]_{\mathfrak{A}}$. Now:

$\mathfrak{A} : \Theta_n$ implies $\mathfrak{A} \models \mathcal{T}_n$ For any $a \in \mathrm{dom}(\mathfrak{A})$, we first verify the left conjunct of $\mathit{PSpec}(a)$: if $f(a) = b$ for some $b \neq a$ and $f \in [\![\mathbf{Stat}_{\mathrm{fun}} \setminus \{\mathsf{parent}\}]\!]_{\mathfrak{A}}$, by definition of an n-BFL we have $p(f(a)) = a$.

For the right conjunct, the definition of an n-BFL also gives $f(p(a)) = a$ for some $f \in [\![\mathbf{Stat}_{\mathrm{fun}} \setminus \{\mathsf{parent}\}]\!]_{\mathfrak{A}}$.

Now we show that for all $0 \leqslant k \leqslant n$, if the longest path away from the root starting from a in $G_{\mathfrak{A}}$ is of length at most k, then a satisfies H^n. By induction on n, the base case is trivial: we have $k = 0$ and a satisfies $H^0(x) = \bigwedge_{f \in \mathbf{f}} f(x) = x$. Otherwise take any child $f(a)$ of a (for some $f \in \mathbf{f}$), the longest path from $f(a)$ is of length at most $k - 1$, so by induction we have $f(a)$ satisfies H^{n-1}. Since this is for any f, a satisfies H^n.

$\mathfrak{A} \models \mathcal{T}_n$ implies $\mathfrak{A} : \Theta_n$ Since any element satisfies H^n, there is no forward path of length $> n$. The last thing to check is that we have a tree, not a DAG. Take $a, b \in \mathrm{dom}(\mathfrak{A})$, $f, g \in [\![\mathbf{Stat}_{\mathrm{fun}} \setminus \{\mathsf{parent}\}]\!]_{\mathfrak{A}}$ such that $a \neq f(a) = g(b) \neq b$. Since $\mathit{PSpec}(a)$ and $\mathit{PSpec}(b)$ hold in $\mathfrak{A}$, the first conjuncts gives us respectively $p(f(a)) = a$ and $p(g(b)) = b$. We assumed $f(a) = g(b)$, so

$a = b$. This also shows that the graph of p contains the parent relation of the loop-free union of the graphs of $[\![\mathbf{Stat}_{\mathrm{fun}} \setminus \{\mathrm{parent}\}]\!]_{\mathfrak{A}}$. To see that the graph of p is contained in it, consider the second conjunct of *PSpec*: whenever p is not the identity on some a, there must be some f such that $f(p(a)) = a$. ☐

Lemma A.0.1. $\cdot \models_{p} \mathsf{Has}_0 \wedge \mathsf{Has}_0^\star$

Proof. Immediate by lemma 5.2.2. ☐

A.1. Building context-free and retractable formulas

Proof of lemma 3.3.1:

Lemma 3.3.1. If ϕ is (I, d)-context-free then ϕ is (I', d')-context-free for any $d' \geqslant d$, $I' \models I$.

Proof. Take any $d' \geqslant d$, $\mathfrak{A}, \mathfrak{B}, \mu : \langle\phi\rangle \to \mathrm{dom}(\mathfrak{A})$ and $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d')) \to \mathfrak{B}$, $\mathfrak{A}, \mu \models \phi$ and $\mathfrak{B} \models \forall(I' \vee m)$. Since $\mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d') \subseteq \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d)$, by lemma 2.4.4, $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d)) \to \mathfrak{B}$; moreover $\mathfrak{B} \models \forall(I' \vee m)$ trivially implies $\mathfrak{B} \models \forall(I \vee m)$ so by assumption on ϕ we get $\mathfrak{B}, m \circ \mu \models \phi$. ☐

Proof of lemma 3.3.2:

Lemma 3.3.2. If ϕ is (I, d)-retractable then ϕ is (I', d')-retractable for any $d' \geqslant d$, $I' \models I$.

Proof. Take any $d' \geqslant d$, $\mathfrak{A}, \mathfrak{B}, \mu : \langle\phi\rangle \to \mathrm{dom}(\mathfrak{A})$ and $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d')) \to \mathfrak{B}$, $\mathfrak{B}, m \circ \mu \models \phi$ and $\mathfrak{B} \models \forall(I \vee m)$. Since $\mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d') \subseteq \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d)$, by lemma 2.4.4, $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d)) \to \mathfrak{B}$; moreover $\mathfrak{B} \models \forall(I' \vee m)$ trivially implies $\mathfrak{B} \models \forall(I \vee m)$ so by assumption on ϕ we get $\mathfrak{A}, \mu \models \phi$. ☐

Proof of lemma 3.3.3

Lemma 3.3.3. A quantifier-free formula ϕ is $(\top, 0)$-context-free.

Proof. Let $\mathfrak{A}, \mu \models \phi$, $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(0)) \to \mathfrak{B}$. Since ϕ is quantifier-free and m is an embedding, $\mathfrak{B}, m \circ \mu \models \phi$. ☐

Proof of lemma 3.3.4

Lemma 3.3.4. A quantifier-free formula ϕ is $(\top, 0)$-retractable.

Proof. Take $d = 0$ as retraction radius. For $\mathfrak{A}, \mathfrak{B}, \mu : \langle\phi\rangle \to \mathrm{dom}(\mathfrak{A})$, $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d)) \to \mathfrak{B}$ with $\mathfrak{B}, m \circ \mu \models \phi$, m is an embedding so $\mathfrak{A}, \mu \models \phi$. ☐

Proof of lemma 3.3.5:

Lemma 3.3.5. If ϕ_1 is (I, d_1)-context-free (resp. retractable) and ϕ_2 is (I, d_2)-context-free (resp. retractable) then $\phi_1 \wedge \phi_2$ is $(I, \max(d_1, d_2))$-context-free (resp. retractable).

Proof. Take $\mathfrak{A}, \mathfrak{B}, \mu : \langle\phi_1 \wedge \phi_2\rangle \to \mathrm{dom}(\mathfrak{A}), m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(\max(d_1, d_2)) \to \mathfrak{B}$ such that $\mathfrak{A}, \mu \models \phi_1 \wedge \phi_2$ (resp. $\mathfrak{B}, m \circ \mu \models \phi_1 \wedge \phi_2$) and $\mathfrak{B} \models \forall(I \vee m)$. We want $\mathfrak{B}, m \circ \mu \models \phi_1 \wedge \phi_2$ (resp. $\mathfrak{A}, \mu \models \phi_1 \wedge \phi_2$).

By lemma 2.4.4, $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}\left(\mu\left(\langle\phi_1\rangle\right) \cup [\![\Delta]\!]_{\mathfrak{A}}, d_1\right) \to \mathfrak{B}$. So by definition of context-free (resp. retractable), $\mathfrak{B}, m \circ \mu \models \phi_1$ (resp. $\mathfrak{A}, \mu \models \phi_1$). Similarly, $\mathfrak{B}, m \circ \mu \models \phi_2$ (resp. $\mathfrak{A}, \mu \models \phi_2$). ☐

Proof of lemma 3.3.6:

Lemma 3.3.6. If ϕ_1 is (I, d_1)-context-free (resp. retractable) and ϕ_2 is (I, d_2)-context-free (resp. retractable) then $\phi_1 \vee \phi_2$ is $(I, \max(d_1, d_2))$-context-free (resp. retractable).

Proof. Take $\mathfrak{A}, \mathfrak{B}, \mu : \langle \phi_1 \vee \phi_2 \rangle \to \mathrm{dom}(\mathfrak{A}), m : (\mathfrak{A}, \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(\max(d_1, d_2)) \to \mathfrak{B}$ such that $\mathfrak{A}, \mu \vDash \phi_1 \vee \phi_2$ (resp. $\mathfrak{B}, m \circ \mu \vDash \phi_1 \vee \phi_2$) and $\mathfrak{B} \vDash \forall (I \vee m)$. We want $\mathfrak{B}, m \circ \mu \vDash \phi_1 \vee \phi_2$ (resp. $\mathfrak{A}, \mu \vDash \phi_1 \vee \phi_2$).

For $i \in \{1, 2\}$, suppose $\mathfrak{A}, \mu \vDash \phi_i$ (resp. $\mathfrak{B}, m \circ \mu \vDash \phi_i$). By lemma 2.4.4, $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}(\mu(\langle \phi_i \rangle) \cup [\![\Delta]\!]_{\mathfrak{A}}, d_i) \to \mathfrak{B}$. So by definition of context-free (resp. retractable), $\mathfrak{B}, m \circ \mu \vDash \phi_i$ (resp. $\mathfrak{A}, \mu \vDash \phi_i$). □

Proof of lemma 3.3.7:

Lemma 3.3.7. If ϕ is (I, d)-context-free and $\alpha(x, y)$ is a guard, then $\forall x.\ \alpha(x, y) \to \phi$ is $(I, d + 1)$-context-free.

Proof. Let $\mathfrak{A}, \mathfrak{B}, \mu : \langle \phi \rangle \setminus \{x\} \to \mathrm{dom}(\mathfrak{A})$ and $m : (\mathfrak{A}, \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d + 1)) \to \mathfrak{B}$ with $\mathfrak{A}, \mu \vDash \forall x.\ \alpha(x, y) \to \phi$ and $\mathfrak{B} \vDash \forall (I \vee m)$. We want $\mathfrak{B}, m \circ \mu \vDash \forall x.\ \alpha(x, y) \to \phi$.

Take any $a \in \mathrm{dom}(\mathfrak{B})$ such that $\mathfrak{B}, m \circ \mu, x \mapsto a \vDash \alpha(x, y)$. Let $b := \mu(y)$. By lemma 3.2.1, $\delta_{\mathfrak{B}}(a, m(b)) \leqslant 1$. Since $b \in \mathrm{Im}(\mu)$, by definition of a match, $a \in m(\mathfrak{A})$. Since m is an embedding, $\mathfrak{A}, x \mapsto m^{-1}(a), y \mapsto b \vDash \alpha(x, y)$. Since $\mathfrak{A}, \mu \vDash \forall x.\ \alpha(x, y) \to \phi$, we get $\mathfrak{A}, \mu, x \mapsto m^{-1}(a) \vDash \phi$. Moreover, by lemma 3.2.1, $\delta_{\mathfrak{A}}(m(a), b) \leqslant 1$, so $\mathcal{B}^{\Delta,\nu}_{\mathfrak{A}}(d) \subseteq \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d + 1)$ where $\nu := \mu, x \mapsto m^{-1}(a)$. We can now apply lemma 2.4.4 to m and get $m : (\mathfrak{A}, \mathcal{B}^{\Delta,\nu}_{\mathfrak{A}}(d)) \to \mathfrak{B}$. Since ϕ is (I, d)-context-free, $\mathfrak{B}, \nu \vDash \phi$. Since we took any $a \in \mathrm{dom}(\mathfrak{B})$, we have $\mathfrak{B}, \mu \vDash \forall x.\ \alpha(x, y) \to \phi$.
□

Proof of lemma 3.3.8:

Lemma 3.3.8. If ϕ is (I, d)-retractable then for any x, $\forall x.\ \phi$ is (I, d)-retractable.

Proof. Let d be a retraction radius for ϕ, I. Let $\mathfrak{A}, \mathfrak{B}, \mu : \langle \phi \rangle \setminus \{x\} \to \mathrm{dom}(\mathfrak{A})$ and $m : (\mathfrak{A}, \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d)) \to \mathfrak{B}$ such that $\mathfrak{B}, m \circ \mu \vDash \forall x.\ \phi$ and $\mathfrak{B} \vDash \forall (I \vee m)$. We want $\mathfrak{A}, \mu \vDash \forall x.\ \phi$.

Take any $a \in \mathrm{dom}(\mathfrak{A})$. Since $\mathfrak{B} \vDash \forall x.\ \phi$, $\mathfrak{B}, m \circ \mu, x \mapsto m(a) \vDash \phi$. Since ϕ is (I, d)-retractable and $m^{-1}(m \circ \mu, x \mapsto m(a)) = \mu, x \mapsto a$, we get $\mathfrak{A}, \mu, x \mapsto a \vDash \phi$. □

Proof of lemma 3.3.9:

Lemma 3.3.9. If ϕ is (I, d)-context-free then for any x, $\exists x.\ \phi$ is (I, d)-retractable.

Proof. Let $\mathfrak{A}$, $\mathfrak{B}, \mu : \langle \phi \rangle \setminus \{x\} \to \mathrm{dom}(\mathfrak{A})$ and $m : (\mathfrak{A}, \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d)) \to \mathfrak{B}$ such that $\mathfrak{A}, \mu \vDash \exists x.\ \phi$ and $\mathfrak{B} \vDash \forall (I \vee m)$. We want $\mathfrak{B}, m \circ \mu \vDash \exists x.\ \phi$.

There is $a \in \mathrm{dom}(\mathfrak{A})$ such that $\mathfrak{A}, \mu, x \mapsto a \vDash \phi$. Since ϕ is (I, d)-context-free, $\mathfrak{B}, m \circ \mu, x \mapsto m(a) \vDash \phi$. □

Proof of lemma 3.3.10:

Lemma 3.3.10. If ϕ is (I, d)-retractable and $\alpha(x, y)$ is a guard then $\exists x.\ \alpha(x, y) \wedge \phi$ is $(I, d + 1)$-retractable.

Proof. Let d be a retraction radius for ϕ, I, let $\mathfrak{A}, \mathfrak{B}, \mu : \langle \phi \rangle \setminus \{x\} \to \mathrm{dom}(\mathfrak{A})$ and $m : (\mathfrak{A}, \mathcal{B}^{\Delta,\mu}_{\mathfrak{A}}(d + 1)) \to \mathfrak{B}$ with $\mathfrak{B}, m \circ \mu \vDash \exists x.\ \alpha(x, y) \wedge \phi$ and $\mathfrak{B} \vDash \forall (I \vee m)$. We want $\mathfrak{A}, \mu \vDash \exists x.\ \alpha(x, y) \wedge \phi$.

There is some $a \in \mathrm{dom}(\mathfrak{B})$ such that $\mathfrak{B}, m \circ \mu, x \mapsto a \vDash \alpha(x, y) \wedge \phi$. Let $b := \mu(y)$. By lemma 3.2.1, $\delta_{\mathfrak{B}}(a, m(b)) \leqslant 1$. Since $b \in \mathrm{Im}(\mu)$, by definition of a match $a \in m(\mathfrak{A})$. Since m is an embedding, $\mathfrak{A}, x \mapsto m^{-1}(a), y \mapsto b \vDash \alpha(x, y)$, so again by lemma 3.2.1, $\delta_{\mathfrak{A}}(m^{-1}(a), b) \leqslant 1$. So $\mathcal{B}_{\mathfrak{A}}^{\Delta,\nu}(d) \subseteq \mathcal{B}_{\mathfrak{A}}^{\Delta,\mu}(d+1)$ where $\nu := \mu, x \mapsto m^{-1}(a)$. We can now apply lemma 2.4.4 to m and get $m : (\mathfrak{A}, \mathcal{B}_{\mathfrak{A}}^{\Delta,\nu}(d)) \to \mathfrak{B}$. Since ϕ is (I, d)-retractable, $\mathfrak{A}, \nu \vDash \phi$. So $\mathfrak{A}, \mu \vDash \exists x.\ \alpha(x, y) \wedge \phi$. $\square$

Index

Bibliography

[Bárány et al., 2011] Bárány, V., Ten Cate, B., and Segoufin, L. (2011). Guarded negation. In *International Colloquium on Automata, Languages, and Programming*, pages 356–367. Springer.

[Bard, 2004] Bard, J. B. L. (2004). Ontologies in biology: design, application and future challenges. *Nature Reviews Genetics*, 5:213–222.

[Bartholomew et al., 2011] Bartholomew, M., Lee, J., and Meng, Y. (2011). First-order extension of the FLP stable model semantics via modified circumscription. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 724.

[Basso-Blandin et al., 2015] Basso-Blandin, A., Fontana, W., and Harmer, R. (2015). A knowledge representation meta-model for rule-based modelling of signalling networks. In *Proceedings of the Eleventh International Workshop on Developments in Computational Models, DCM 2015, Cali, Colombia, October 28, 2015.*, pages 47–59.

[Boniolo et al., 2010] Boniolo, G., D'Agostino, M., and Di Fiore, P. P. (2010). Zsyntax: A formal language for molecular biology with projected applications in text mining and biological prediction. *PLOS ONE*, 5(3):1–12.

[Börger et al., 1997] Börger, E., Grädel, E., and Gurevich, Y. (1997). *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer.

[Boutillier et al., 2017] Boutillier, P., Ehrhard, T., and Krivine, J. (2017). Incremental update for graph rewriting. In *Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, pages 201–228.

[Boutillier et al., 2018] Boutillier, P., Maasha, M., Li, X., Medina-Abarca, H. F., Krivine, J., Feret, J., Cristescu, I., Forbes, A. G., and Fontana, W. (2018). The Kappa platform for rule-based modeling. *Bioinformatics*, 34(13):583–592.

[Chaudhri et al., 2014] Chaudhri, V. K., Elenius, D., Goldenkranz, A., Gong, A., Martone, M. E., Webb, W., and Yorke-Smith, N. (2014). Comparative analysis of knowledge representation and reasoning requirements across a range of life sciences textbooks. *Journal of Biomedical Semantics*, 5:51.

[Chylek et al., 2014] Chylek, L., Harris, L., Tung, C.-S., Faeder, J., Lopez, C., and Hlavacek, W. (2014). Rule-based modeling (...). *Wiley interdisciplinary reviews Systems biology and medicine*, 6(1):13–36.

[Cohen, 2015] Cohen, P. R. (2015). DARPA's big mechanism program. *Physical Biology*, 12(4):045008.

[Conradie, 2006] Conradie, W. (2006). On the strength and scope of DLS. *Journal of Applied Non-Classical Logics*, 16(3-4):279–296.

[Danos et al., 2012] Danos, V., Feret, J., Fontana, W., Harmer, R., Hayman, J., Krivine, J., Thompson-Walsh, C., and Winskel, G. (2012). Graphs, rewriting and pathway reconstruction for rule-based models. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[Danos et al., 2007] Danos, V., Feret, J., Fontanta, W., Harmer, R., and Krivine, J. (2007). Rule based modeling of biological signaling. In Caires, L. and Vasconcelos, V. T., editors, *Proceedings of CONCUR 2007*, volume 4703 of *LNCS*, pages 17–41. Springer.

[De Kleer and Konolige, 1989] De Kleer, J. and Konolige, K. (1989). Eliminating the fixed predicates from a circumscription. *Artificial Intelligence*, 39(3):391–398.

[Demir et al., 2010] Demir, E., P Cary, M., Paley, S., Fukuda, K., Lemer, C., and Vastrik, I. (2010). The BioPAX community standard for pathway data sharing. *Nature Biotechnology*, 28.

[Despeyroux, 2016] Despeyroux, J. (2016). (Mathematical) logic for systems biology. In *CMSB*, pages 3–12. Springer.

[Doherty et al., 2004] Doherty, P., Kertes, S., Magnusson, M., and Szalas, A. (2004). Towards a logical analysis of biochemical pathways. In *European Workshop on Logics in Artificial Intelligence*, pages 667–679. Springer.

[Doherty et al., 1997] Doherty, P., Łukaszewicz, W., and SzaŁas, A. (1997). Computing circumscription revisited: A reduction algorithm. *Journal of Automated Reasoning*, 18(3):297–336.

[Doherty et al., 1998] Doherty, P., Łukaszewicz, W., and Szałas, A. (1998). General domain circumscription and its effective reductions. *Fundamenta Informaticae*, 36(1):23–55.

[Eiter et al., 1993] Eiter, T., Gottlob, G., and Gurevich, Y. (1993). Curb your theory! a circumscriptive approach for inclusive interpretation of disjunctive information. In *Proceedings of IJCAI'93*, volume 13, pages 634–634.

[Eker et al., 2002] Eker, S., Knapp, M., Laderoute, K., Lincoln, P., Meseguer, J., and Sonmez, K. (2002). Pathway logic: symbolic analysis of biological signaling. *Biocomputing*, pages 400–412.

[Etherington, 1986] Etherington, D. W. (1986). *Reasoning with incomplete information : investigations of non-monotonic reasoning*. PhD thesis, University of British Columbia.

[Faeder et al., 2009] Faeder, J. R., Blinov, M. L., and Hlavacek, W. S. (2009). *Rule-Based Modeling of Biochemical Systems with BioNetGen*, pages 113–167. Humana Press, Totowa, NJ.

[Ferraris et al., 2007] Ferraris, P., Lee, J., and Lifschitz, V. (2007). A new perspective on stable models. In *IJCAI*, volume 7, pages 372–379.

[Ferraris et al., 2011] Ferraris, P., Lee, J., and Lifschitz, V. (2011). Stable models and circumscription. *Artificial Intelligence*, 175(1):236.

[Fisher and Henzinger, 2007] Fisher, J. and Henzinger, T. A. (2007). Executable cell biology. *Nature Biotechnology*, 25(11):1239–1249.

[Fontana and Buss, 1996] Fontana, W. and Buss, L. (1996). The barrier of objects: From dynamical systems to bounded organizations. Working papers, International Institute for Applied Systems Analysis.

[Grädel, 1999] Grädel, E. (1999). On the restraining power of guards. *J. Symb. Log.*, 64(4):1719–1742.

[Gulwani et al., 2017] Gulwani, S., Polozov, O., Singh, R., et al. (2017). Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119.

[Harel et al., 2001] Harel, D., Kozen, D., and Tiuryn, J. (2001). Dynamic logic. In *Handbook of philosophical logic*, pages 99–217. Springer.

[Hodkinson, 2002] Hodkinson, I. (2002). Loosely guarded fragment of first-order logic has the finite model property. *Studia Logica*, 70(2):205–240.

[Hunter, 1990] Hunter, L. (1990). Artificial intelligence and molecular biology. *AI magazine*, 2:27–36.

[Husson and Krivine, 2019] Husson, A. and Krivine, J. (2019). A tractable logic for molecular biology. In Bogaerts, B., Erdem, E., Fodor, P., Formisano, A., Ianni, G., Inclezan, D., Vidal, G., Villanueva, A., Vos, M. D., and Yang, F., editors, Proceedings 35th International Conference on *Logic Programming* (Technical Communications) , Las Cruces, NM, USA, September 20-25, 2019, volume 306 of *Electronic Proceedings in Theoretical Computer Science*, pages 101–113. Open Publishing Association.

[Krivine, 2017] Krivine, J. (2017). Systems biology. *SIGLOG News*, 4(3):43–61.

[Laurent et al., 2018] Laurent, J., Medina-Abarca, H. F., Boutillier, P., Yang, J., and Fontana, W. (2018). A trace query language for rule-based models. In *International Conference on Computational Methods in Systems Biology*, pages 220–237. Springer.

[Lazebnik, 2002] Lazebnik, Y. (2002). Can a biologist fix a radio? – or, what I learned while studying apoptosis. *Cancer Cell*, 2(3):179–182.

[Libkin, 2013] Libkin, L. (2013). *Elements of finite model theory*. Springer Science & Business Media.

[Mamouras, 2016] Mamouras, K. (2016). Synthesis of strategies using the hoare logic of angelic and demonic nondeterminism. *arXiv preprint arXiv:1606.09110*.

[Mason et al., 1999] Mason, C. S., Springer, C. J., Cooper, R. G., Superti-Furga, G., Marshall, C. J., and Marais, R. (1999). Serine and tyrosine phosphorylations cooperate in raf-1, but not b-raf activation. *The EMBO journal*, 18(8):2137–2148.

[McCarthy, 1980] McCarthy, J. (1980). Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1):27 – 39. Special Issue on Non-Monotonic Logic.

[McCarthy, 1986] McCarthy, J. (1986). Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28(1):89 – 116.

[Miller and Sirán, 2005] Miller, M. and Sirán, J. (2005). Moore graphs and beyond: A survey of the degree/diameter problem. *The electronic journal of combinatorics*, 1000:DS14–Dec.

[Nonnengart et al., 1999] Nonnengart, A., Ohlbach, H. J., and Szałas, A. (1999). Elimination of predicate quantifiers. In *Logic, Language and Reasoning*, pages 149–171. Springer.

[Novère, 2015] Novère, N. L. (2015). Quantitative and logic modelling of molecular and gene networks. *Nature Reviews Genetics*, 16:146–158.

[Reiter, 1991] Reiter, R. (1991). The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy*, 27:359–380.

[Reiter, 2001] Reiter, R. (2001). *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT press.

[Stevens et al., 2000] Stevens, R., Goble, C. A., and Bechhofer, S. (2000). Ontology-based knowledge representation for bioinformatics. *Briefings in Bioinformatics*, 1(4):398–414.

[UniProt Consortium, 2019] UniProt Consortium, T. (2019). Uniprotkb - p09560 (raf1_xenla) [uniprot.org/uniprot/p09560].

[UniProt Consortium, 2018] UniProt Consortium, T. (2018). UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 46(5):2699–2699.

[Wan et al., 2014] Wan, H., Xiao, Z., Yuan, Z., Zhang, H., and Zhang, Y. (2014). Computing general first-order parallel and prioritized circumscription. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1105–1111. AAAI Press.

[Yaman et al., 2018] Yaman, F., Adler, A., and Beal, J. (2018). AI challenges in synthetic biology engineering. In *The Thirtieth AAAI Conference on Innovative Applications of Artificial Intelligence (IAAI-18)*.

[Yang et al., 2011] Yang, Q., You, J.-H., and Feng, Z. (2011). Integrating rules and description logics by circumscription. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11. AAAI PRess.

[Zhang and Ying, 2010] Zhang, H. and Ying, M. (2010). Decidable fragments of first-order language under stable model semantics and circumscription. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, pages 375–380. AAAI Press.