Technical Report http://hdl.handle.net/2142/32952, Jul 2012

Reachability Logic

Grigore Roşu Andrei Ştefănescu
University of Illinois, USA
{grosu, stefane1}@illinois.edu

Ştefan Ciobâcă
University "Alexandru Ioan Cuza", Romania
stefan.ciobaca@info.uaic.ro

Brandon M. Moore
University of Illinois, USA
bmmoore@illinois.edu

Abstract

This paper introduces *reachability logic*, a language-independent seven-rule proof system for deriving reachability properties of systems. The key ingredients of reachability logic are its sentences, which are called *reachability rules* and generalize the transitions of operational semantics and the Hoare triples of axiomatic semantics, and the *Circularity* proof rule, which generalizes invariant proof rules for iterative and recursive constructs in axiomatic semantics. The target transition system is described as a set of reachability rules, which are taken as axioms in a reachability logic proof. Typical definition styles which can be read as collections of reachability rules include conventional small-step and big-step operational semantics. The reachability logic proof system is shown sound (in the sense of partial correctness) and relatively complete. The soundness result has also been formalized in Coq, allowing to convert reachability logic proofs into proof certificates depending only on the operational semantics and the unavoidable domain reasoning. Reachability logic thus eliminates the need to independently define an axiomatic and an operational semantics for each language, and the non-negligible effort to prove the former sound and complete w.r.t the latter.

1. Introduction

Operational semantics are easier to define and understand than other semantics, because they can be thought of as formal implementations of the languages they define. For example, a big-step semantics can be thought of as a recursive interpreter, while a small-step semantics as an execution engine describing each computational step that can be performed at each moment. Operational semantics typically require little mathematical knowledge or formal training, which make them common introductory topics in programming language courses. Moreover, operational semantics scale and, being executable and thus testable, yield trusted reference models/implementations for the defined languages. For example, in the case of the C language, the operational semantics in [3] is used to develop certifying compilers, the one in [1] to show axiomatic semantics sound, the one in [8] to check undefinedness and to formally analyze programs, etc.

In spite of all the advantages above, operational semantics are typically considered inappropriate for formal program reasoning. The main reason is that proofs based on operational semantics tend to be rather low-level and tedious, having to formalize and then work directly with the corresponding transition system. Existing program reasoning approaches, such as Hoare logic or dynamic logic, require (re)defining the target language as a set of abstract proof rules, which often require non-trivial program transformations (e.g., to eliminate the side effects from expressions) and are therefore often hard to understand and trust. Indeed, the state-of-the-art in mechanical verification is to develop and prove such language-specific proof systems sound with respect to more trusted semantics [1, 12, 17, 19, 26, 36]. While defining a language twice or more using different semantic approaches and then formally proving the relationships between them can in theory help with finding errors in each semantics, in practice this is quite uneconomical, because languages evolve and some require hundreds or even thousands of semantic rules, and manuals are not always rigorous (so the same mistake is possible in all semantics). In our experience defining operational semantics for real languages like C [8], Java (1.4) [9], Verilog [22], etc., the capability to execute semantics on thousands of programs (e.g., benchmarks used to test compilers) is a quite effective means to catch semantic errors. Ideally, we would like an effective way to use operational semantics for program reasoning and verification, so that defining alternative semantics for the same programming language becomes an option instead of a necessity.

Recent work, although limited to operational semantics defined with rules without premises, suggests that effective program verification directly based on formal operational semantics *is possible*. A proof system for reachability specifications was introduced and shown sound (i.e., partially correct) in [33], derivations use the unconditional operational semantics rules unchanged, as axioms; then [32] shows that, for a simple imperative language, any Hoare logic proof can be mechanically translated into a reachability proof of similar size, thus (1) showing that nothing is lost in terms of expressiveness or compactness, and (2) indirectly proving the (relative) completeness of the reachability proof system for a simple but standard language. A slightly modified reachability proof system is given in [34] and proved both sound and complete, together with an implementation of it into a practical program verifier based on a fragment of C, called MatchC. As already mentioned, unfortunately all the above require that the operational semantics be defined using *unconditional* rules. While this requirement is acceptable for some operational semantics styles, such as reduction semantics with evaluation contexts [38], it unfortunately excludes the two most standard operational semantics approaches, namely Plotkin's small-step [29] and Kahn's big-step [5] semantics, as well as combinations.

Inspired by the recent work described above and challenged by its limitations, in this paper we propose a novel proof system for reachability, referred to as *reachability logic*. The proof system can also take as axioms operational semantics that make use of *conditional* rules, and can derive any reachability property of the language expressible as an unconditional reachability rule. Typically, Hoare triples are instances of unconditional reachability rules. By supporting conditional rules as axioms, our new proof system works with virtually all the operational semantics approaches (details are given in Section 2). A conditional reachability rule (formally introduced in Section 4), is a sentence of the form

$$\varphi \Rightarrow \varphi' \text{ if } \varphi_1 \Rightarrow \varphi_1' \wedge \cdots \wedge \varphi_n \Rightarrow \varphi_n'$$

where $\varphi, \varphi', \varphi_1, \varphi_1', ..., \varphi_n, \varphi_n'$ are *matching logic patterns*. A matching logic pattern specifies structural properties of the program configuration by means of special predicates, namely configuration terms with variables, whose satisfaction is given by "matching". In the case of a simple C-like imperative language defined using a big-step semantics with configurations $\langle \mathtt{code},\sigma\rangle$, $\langle i\rangle$ and $\langle\sigma\rangle$, where $\mathtt{code}$ ranges over program syntax, i over integers and σ over states, the following conditional reachability rule defines the semantics of the positive case of the conditional statement:

$$\langle \texttt{if}\, e\ s_1\ s_2,\ \sigma\rangle \wedge i \neq 0 \Rightarrow \langle\sigma'\rangle \ \text{ if }\ \langle e,\ \sigma\rangle \Rightarrow \langle i\rangle \bigwedge \langle s_1,\ \sigma\rangle \Rightarrow \langle\sigma'\rangle$$

It says that concrete program configuration γ matching pattern $\langle \texttt{if}\, e\ s_1\ s_2,\sigma\rangle \wedge i \neq 0$ with witness valuation ρ, i.e., $\rho(\langle \texttt{if}\, e\ s_1\ s_2,\sigma\rangle) = \gamma$ and $\rho(i) \neq 0$, reduces to configuration $\langle\rho(\sigma')\rangle$, provided $\langle\rho(e), \rho(\sigma)\rangle$ reduces to $\langle\rho(i)\rangle$ and $\langle\rho(s_1), \rho(\sigma)\rangle$ reduces to $\langle\rho(\sigma')\rangle$.

On the other hand, if SUM is the program summing up in s all the numbers up to n, then the unconditional reachability rule

$$\langle \texttt{SUM}, (\texttt{s} \mapsto s, \texttt{n} \mapsto n)\rangle \wedge n \geq 0 \ \Rightarrow\ \langle(\texttt{s} \mapsto n * (n+1)/2, \texttt{n} \mapsto 0)\rangle$$

states the desired property of SUM. In words, if we execute the configuration holding the program SUM and a state binding program variables s and n to integers s and $n \geq 0$ using the big-step semantics, then we reach a configuration holding a state that binds program variable s to the sum of numbers up to n and n to 0, respectively.

Our new language-independent seven-rule proof system (shown in Figure 3) can formally derive unconditional reachability rules like the one for SUM above using, as axioms, operational semantics given as sets of conditional reachability rules like the one for the if statement above. Five of the proof rules provide the formal machinery needed to perform symbolic execution using the operational semantics rules, including means to perform domain reasoning in order to make rules match and to split into cases when multiple rules match. The two special rules are *Abstraction* and *Circularity*. Abstraction allows to abstract away irrelevant details, using existential quantifiers. Circularity is a language-independent generalization of the typical language-specific invariant proof rules associated to language constructs with repetitive behaviors, such as loops, recursive functions, jumps, etc.

By generalizing the basic elements of both operational and axiomatic semantics, reachability logic smoothly unifies the two in a more general framework. Indeed, one can use conditional reachability rules to define operational semantics which do not necessarily correspond to any particular operational style (for example, big-step for expressions, reduction semantics with evaluation contexts for statements, etc.), and then derive reachability rules which do not necessarily correspond to any particular Hoare triple.

Contributions. This paper makes the following contributions:

1. It introduces the *conditional reachability rule*, which generalizes the previous unconditional reachability rule to allow capturing the various small-step and big-step operational semantic rules.
2. It introduces *reachability logic*, a seven-rule proof system that takes a set of conditional reachability rules (e.g., an operational semantics) as axioms and derives unconditional reachability rules (e.g., ones corresponding to Hoare triples).
3. It proves the proof system *sound*, that is, partially correct. Due to its practical relevance in producing proof objects, the soundness result has also been formalized and proved in Coq.
4. It proves the proof system *relatively complete*. This result is significantly more powerful than the relative completeness of Hoare logic, because it is proved once for all languages, rather than separately for each language.

The supporting code and Coq mechanical proofs can be found at `http://fsl.cs.uiuc.edu/RL`.

Related work. We fully adhere to the fundamental philosophy of the unified theory of programming initiative [16] and of the mechanical verification community to reduce the correctness of program verification to a trusted formal semantics of the target language [1, 12, 17, 19, 26, 36], although our methods are somewhat different. Instead of developing a framework where various semantic approaches coexists with systematic relationships between them, which requires us to still provide two or more semantics of the same language, we advocate for a framework where we need only *one* semantics of the language, which is operational, with the underlying theory providing the necessary machinery to achieve the benefits of other individual semantics without the additional costs.

To regard a program as a specification transformer to analyze programs in a forwards-style goes back to Floyd [11]. However, his rules are not concerned with structural configurations, are not meant to be operational, and introduce quantifiers. Similar ideas have been used in equational algebraic specifications of programming languages [14] and in evolving specifications [28]. Conceptually, what distinguishes our approach from these is the use of matching logic to specify configurations of interests by means of patterns, which give access to all the structural details. The use of variables in patterns offers a comfortable level of abstraction by mentioning in each rule only the necessary configuration components.

Dynamic logic [15] adds modal operators to FOL to embed program fragments within specifications. For example, $\psi \rightarrow [\texttt{s}]\psi'$ means "after executing s in a state satisfying ψ, a state may be reached which satisfies ψ'". In matching logic, programs and specifications also coexits in the same logic, but we use it only for expressing static properties. We express the dynamic properties using reachability logic which, unlike dynamic logic which still requires language-specific proof rules, consists of only language-independent proof rules and works with any operational semantics.

Separation logic [27, 30] has been proposed as a state specification formalism to enhance Hoare logic in the presence of heaps. While reachability logic is an alternative to, rather than an extension of Hoare logic, one could use separation logic as a pattern specification formalism. This is not precluded by presenting our results in terms of matching logic, as separation logic has been shown to be a matching logic instance when the configuration model is chosen to be that of heaps [34]. However, operational semantics of complex language require many more other configuration components besides the heap, and matching logic has been designed to work with arbitrarily complex configurations.

Our own related work. Devising a sound, relatively complete and language-independent proof system that works with conditional rules as axioms was the holy-grail of our research over the last several years. We have thus made several prior attempts in this direction. First-order matching logic and a first proof system based on it was presented in [35]. However, that proof system was language-specific and not parametric in an operational semantics like the one in this paper. An early implementation of MatchC, based on that proof system, was presented in [31]. Our first language-independent proof system can be found in [33], and consists of nine proof rules. In [32] we showed its relative completeness for a particular simple imperative language, by providing a mechanical translation of Hoare logic proof derivations into derivations using that proof system. Finally, [34] presents a more elegant, eight-rule proof system for reachability, which we use as a basis here, together with language-independent proofs of its soundness and relative completeness, and together with a reimplementation of MatchC based on it. However, until now we were only able to support very particular operational semantic definitions, namely ones with unconditional rules. While all the above are clearly inferior to our new proof system and its language-independent proofs of soundness and completeness, they were nevertheless crucial milestones.

2. Operational Semantics using Rewriting

There are many operational semantics approaches. Since our goal is to develop a reachability reasoning framework that works for all operational semantics, a first challenge is to find a framework that uniformly and formally supports the variety of operational semantics. One possibility is to pick a rich framework, e.g., higher-order logic [18], the calculus of constructions [6], or rewriting logic [23], which provides all the needed mathematical infrastructure (and much more). [1] However, we prefer to present our results following a different approach here, which we believe makes them more widely accessible. Specifically, we pick a weak framework capable of expressing operational semantics, namely (a fragment of conditional) *term rewriting*. Rewriting is so basic that any rich framework can express it; some frameworks have builtin support for rewriting, others can express it by means of proof strategies, others can define it as a transitive binary relation, etc.

In this paper we assume that operational semantics can be defined with particular conditional rewrite rules of the form

$$cfg \Rightarrow cfg' \text{ if } b \bigwedge cfg_1 \Rightarrow cfg'_1 \bigwedge b_1 \bigwedge \ldots \bigwedge cfg_n \Rightarrow cfg'_n \bigwedge b_n$$

where cfg, cfg', cfg_1, cfg'_1, ..., cfg_n, cfg'_n are configuration terms and b, b_1, ..., b_n are Boolean terms (incrementally) constraining the variables that appear in the configuration terms. For example,

$$\langle \texttt{if}\, e\ s_1\ s_2,\ \sigma\rangle \Rightarrow \langle\sigma'\rangle \text{ if } \langle e,\ \sigma\rangle \Rightarrow \langle i\rangle \bigwedge i \neq 0 \bigwedge \langle s_1,\ \sigma\rangle \Rightarrow \langle\sigma'\rangle$$

defines the big-step operational semantics of the positive case of `if` in a simple C-like language, whose configurations are pairs $\langle \texttt{code},\sigma\rangle$ of a fragment of program and a state mapping program variables to integers, with result configurations $\langle\sigma\rangle$ for statements and $\langle i\rangle$ for expressions (i is an integer). In this particular rule, $n = 2$ and b and b_2 are *true*, so not written. The e, s_1 and s_2, σ and σ', and i, are variables of sorts expression, statement, state, and resp. integer.

The meaning of $\Rightarrow$ is *reachability*, i.e., zero, one or more steps. The rewrites $cfg_i \Rightarrow cfg'_i$ in the condition of a rewrite rule are called "premises", while the Boolean conditions b and b_i are called "side conditions". Theoretically, the order of premises and side conditions is irrelevant, and thus the side conditions could all be merged into one. However, for performance, rewrite engines check the premises and side conditions from left-to-right and users often rely on it.

Our rewrite rules are purposely simple; e.g., since they contain only configuration and Boolean terms, we implicitly consider only top-most rewriting. Like in other rewrite frameworks (e.g., rewriting logic [23]), $\Rightarrow$ is inherently transitively-closed. However, some operational semantics need to express precisely one step, written $\Rightarrow^1$. This can be expressed using the general rewrite relation $\Rightarrow$ [7, 24]. In [24] the one-step transition is eliminated by replacing $\langle \texttt{code},\sigma\rangle \Rightarrow^1 \langle \texttt{code}',\sigma'\rangle$ with $[\texttt{code},\sigma] \Rightarrow \{\texttt{code}',\sigma'\}$ in the conclusion and conditions of rules, where [_, _] and {_, _} are new configuration constructs, and adding one new conditional rule $\langle \texttt{code},\sigma\rangle \Rightarrow \langle \texttt{code}',\sigma'\rangle$ if $[\texttt{code},\sigma] \Rightarrow \{\texttt{code}',\sigma'\}$ to embed $\Rightarrow^1$ into the general reflexively, transitively closed $\Rightarrow$. Hence, $\Rightarrow^1$ is technically unnecessary; conditional rules using just $\Rightarrow$ suffice.

Figure 1 shows a small-step and a big-step operational semantics of a simple imperative language, called IMP, using rewrite rules. In the reminder of the paper, we will refer to three IMP programs:

```
SUM  ≡  s := 0; while (n > 0) (s := s+n; n := n-1)
SUM' ≡  s := 0; while (n > 0)  s := s+n
SUM∞ ≡  n := 1; while (n > 0)  s := s+n
```

`SUM` always terminates, `SUM'` only terminates when $\texttt{n} \leq 0$, and $\texttt{SUM}_\infty$ never terminates. In the small-step semantics, nontermination is represented by infinite "horizontal" computation: each rule application terminates, but there are infinitely many rule applications. In the big-step semantics, nontermination is represented by infinite "vertical" computation: a rule application does not terminate since it requires

[1] We are currently using such a framework, Coq [21], to formalize operational semantics as transition systems and collections of reachability rules, to prove equivalences between such semantics for the same language, to mechanize the soundness of our reachability logic proof system, and to produce proof certificates for verified programs. We are also using the rewriting logic engine Maude [4] to efficiently execute operational semantics defined using rewriting. See `http://fsl.cs.uiuc.edu/`RL for details and links to code.

IMP syntax

PVar	::=	identifiers to be used as program variables
Exp	::=	*PVar* | *Int* | *Exp* + *Exp* | ...
Stmt	::=	`skip` | *PVar* := *Exp* | *Stmt* ; *Stmt*
	|	`if` *Exp Stmt Stmt* | `while` *Exp Stmt*

IMP small-step semantics

$\mathbf{+}_1$	$\langle e_1 + e_2,\ \sigma\rangle \Rightarrow^1 \langle e'_1 + e_2,\ \sigma\rangle$ if $\langle e_1,\ \sigma\rangle \Rightarrow^1 \langle e'_1,\ \sigma\rangle$
$\mathbf{+}_2$	$\langle i_1 + e_2,\ \sigma\rangle \Rightarrow^1 \langle i_1 + e'_2,\ \sigma\rangle$ if $\langle e_2,\ \sigma\rangle \Rightarrow^1 \langle e'_2,\ \sigma\rangle$
$\mathbf{+}_3$	$\langle i_1 + i_2,\ \sigma\rangle \Rightarrow^1 \langle i_1\ +_{Int}\ i_2,\ \sigma\rangle$
lookup	$\langle x,\ \sigma\rangle \Rightarrow^1 \langle \sigma(x),\ \sigma\rangle$ if $x \in Dom(\sigma)$
$\mathbf{asgn}_1$	$\langle x \mathrel{\texttt{:=}} e,\ \sigma\rangle \Rightarrow^1 \langle x \mathrel{\texttt{:=}} e',\ \sigma\rangle$ if $\langle e,\ \sigma\rangle \Rightarrow^1 \langle e',\ \sigma\rangle$
$\mathbf{asgn}_2$	$\langle x \mathrel{\texttt{:=}} i,\ \sigma\rangle \Rightarrow^1 \langle \texttt{skip},\ \sigma[x \leftarrow i]\rangle$ if $x \in Dom(\sigma)$
$\mathbf{seq}_1$	$\langle s_1 \texttt{;} s_2,\ \sigma\rangle \Rightarrow^1 \langle s'_1 \texttt{;} s_2,\ \sigma'\rangle$ if $\langle s_1,\ \sigma\rangle \Rightarrow^1 \langle s'_1,\ \sigma'\rangle$
$\mathbf{seq}_2$	$\langle \texttt{skip;} s_2,\ \sigma\rangle \Rightarrow^1 \langle s_2,\ \sigma\rangle$
$\mathbf{cond}_1$	$\langle \texttt{if}\, e\ s_1\ s_2,\ \sigma\rangle \Rightarrow^1 \langle \texttt{if}\, e'\ s_1\ s_2,\ \sigma\rangle$ if $\langle e,\ \sigma\rangle \Rightarrow^1 \langle e',\ \sigma\rangle$
$\mathbf{cond}_2$	$\langle \texttt{if}\, i\ s_1\ s_2,\ \sigma\rangle \Rightarrow^1 \langle s_1,\ \sigma\rangle$ if $i \neq 0$
$\mathbf{cond}_3$	$\langle \texttt{if}\, 0\ s_1\ s_2,\ \sigma\rangle \Rightarrow^1 \langle s_2,\ \sigma\rangle$
while	$\langle \texttt{while}\, e\ s,\ \sigma\rangle \Rightarrow^1 \langle \texttt{if}\, e\ (s\texttt{;while}\, e\ s)\ \texttt{skip},\ \sigma\rangle$

IMP big-step semantics

+	$\langle e_1{+}e_2,\sigma\rangle \Rightarrow \langle i_1 +_{Int} i_2\rangle$ if $\langle e_1,\sigma\rangle \Rightarrow \langle i_1\rangle \bigwedge \langle e_2,\sigma\rangle \Rightarrow \langle i_2\rangle$
int	$\langle i,\ \sigma\rangle \Rightarrow \langle i\rangle$
lookup	$\langle x,\ \sigma\rangle \Rightarrow \langle \sigma(x)\rangle$ if $x \in Dom(\sigma)$
skip	$\langle \texttt{skip},\ \sigma\rangle \Rightarrow \langle\sigma\rangle$
asgn	$\langle x \mathrel{\texttt{:=}} e,\ \sigma\rangle \Rightarrow \langle \sigma[x \leftarrow i]\rangle$ if $x \in Dom(\sigma) \bigwedge \langle e,\ \sigma\rangle \Rightarrow \langle i\rangle$
seq	$\langle s_1; s_2, \sigma\rangle \Rightarrow \langle\sigma_2\rangle$ if $\langle s_1, \sigma\rangle \Rightarrow \langle\sigma_1\rangle \bigwedge \langle s_2, \sigma_1\rangle \Rightarrow \langle\sigma_2\rangle$
$\mathbf{cond}_1$	$\langle \texttt{if}\, e\ s_1\ s_2, \sigma\rangle \Rightarrow \langle\sigma'\rangle$ if $\langle e,\sigma\rangle \Rightarrow \langle i\rangle \bigwedge i{\neq}0 \bigwedge \langle s_1,\sigma\rangle \Rightarrow \langle\sigma'\rangle$
$\mathbf{cond}_2$	$\langle \texttt{if}\, e\ s_1\ s_2, \sigma\rangle \Rightarrow \langle\sigma'\rangle$ if $\langle e,\ \sigma\rangle \Rightarrow \langle 0\rangle \bigwedge \langle s_2,\ \sigma\rangle \Rightarrow \langle\sigma'\rangle$
$\mathbf{while}_1$	$\langle \texttt{while}\, e\ s,\ \sigma\rangle \Rightarrow \langle\sigma\rangle$ if $\langle e,\ \sigma\rangle \Rightarrow \langle 0\rangle$
$\mathbf{while}_2$	$\langle \texttt{while}\, e\ s,\ \sigma\rangle \Rightarrow \langle\sigma'\rangle$ if $\langle e,\sigma\rangle \Rightarrow \langle i\rangle \bigwedge i{\neq}0 \bigwedge \langle \texttt{s;while}\, e\ s,\ \sigma\rangle \Rightarrow \langle\sigma'\rangle$

Figure 1. The IMP language: syntax, a small-step and a big-step operational semantics. The operational semantics contain rewrite rules making use of ordinary first-order variables: $e, e', e_1, e'_1, e_2, e'_2$ are variables of sort *Exp*; σ, σ' are variables of sort *State*; i, i_1, i_2 are variables of sort *Int*; x is a variable of sort *PVar*; s, s_1, s'_1, s_2 are variables of sort *Stmt*; *code*, *code'* are variables of sort *Exp* or *Stmt*. The underlying mathematical domain is assumed to provide all the needed operations, for example $+_{Int}$, $*_{Int}$, $<_{Int}$, etc., for integers, and $\sigma(x)$, $\sigma[x \leftarrow i]$, $x \in Dom(\sigma)$, etc., for maps.

another rule application to solve one of its premises, which requires another rule application to solve one of its premises, and so on.

We can easily obtain variations of the two semantics in Figure 1, e.g., ones using big-step for expressions and small-step for statements. In fact, as shown in [7], rewrite rules can support virtually all operational semantics styles, including reduction semantics with evaluation contexts [38], the chemical abstract machine [2], continuation-based semantics [10], etc. For reduction semantics with evaluation contexts, the idea in [7] is to add a new configuration construct, _[_], taking a context term and a code fragment term, and define the split and plug operations by means of rewrite rules. In the case of IMP, we would add rules like $C[e_1 + e_2] \Rightarrow C[\square + e_2][e_1]$ if $e_1 \notin Int$ and $C[\square + e_2][i_1] \Rightarrow C[i_1 + e_2]$ in addition to actual semantic rules like $C[i_1{+}i_2] \Rightarrow^1 C[i_1 +_{Int} i_2]$. Additional machinery to compose/decompose contexts is needed, but can also be defined using rules. The reader interested in how our reachability reasoning results apply to operational semantics using evaluation contexts is referred to [32–34] for more details.

Besides theoretical simplicity and uniformity, an additional advantage of using rewriting to define operational semantics is that they can be executed and tested with off-the-shelf rewrite engines. In our experience with Maude [4], rewrite-based operational semantics can stay within an order of magnitude slower than custom, compiled interpreters for the defined languages. This is in our view acceptable, because the interpreters obtained by executing the rewrite-based operational semantics are free and correct-by-construction.

In conclusion, rewrite rules can be used to formally and uniformly define operational semantics. Moreover, they also yield relatively efficient reference models/implementations for the defined languages at no additional cost. The rest of the paper is dedicated to showing that rewrite-based operational semantics are also sufficient for program reasoning; no other (axiomatic) semantics is needed.

3. Matching Logic

Matching logic is a logic for defining and reasoning about structure, especially program configurations, which can include code stacks, heaps, I/O buffers, etc. Matching logic is parametric in a model of configurations. Configurations can be as simple as pairs $\langle \texttt{code}, \sigma \rangle$ with `code` a program fragment and σ a state mapping program variables to integers if one wants to reason about simple imperative languages like IMP (see Section 2), or even as simple as "heap" singletons holding a map from locations to integers if one wants to reason only about heap structures (see [34] for how to capture separation logic as a matching logic instance). Other configurations can be as complex as that of C [8], which has about 70 semantic components.

A matching logic instance defines *configurations*, the concrete structures of interest, *patterns*, which serve as structure abstractions or specifications, and *pattern matching*, a satisfaction relation between configurations and patterns. Until now, matching logic has only been defined in a first-order logic setting. In this paper we generalize matching logic by parameterizing it also in its patterns and in its pattern matching relation. These parameters must respect some mild restrictions and relationships, which will suffice to prove the soundness of our proof system in Section 5.

Using this generalization of matching logic in the proof system for reachability has two benefits, one practical and one theoretical. Practically, it allows our proof system to be used in settings where program properties are stated and reasoned about in other formalisms than first-order matching logic, for example in variants of separation logic, monadic second-order logic, etc. Theoretically, it shows the weakest conditions on a program state logic under which we could show our proof system sound.

3.1 First-Order Matching Logic

Here we recall basic notions of first-order matching logic from [32–35] (there called just "matching logic"), which can be framed as a methodological fragment of first-order logic (FOL) in a given model of configurations (see Section 6.1).

The reader is assumed familiar with basic concepts of algebraic specification and first-order logic (see, e.g., the CASL and Maude systems and their manuals [4, 25], which also provide many examples). Given an *algebraic signature* Σ, we let T_Σ denote the *initial* Σ*-algebra* of ground terms (i.e., terms without variables) and let $T_\Sigma(Var)$ denote the *free* Σ*-algebra* of terms with variables in *Var*. $T_{\Sigma,s}(Var)$ is the set of Σ-terms of sort s. Maps $\rho : Var \to \mathcal{T}$ with $\mathcal{T}$ a Σ-algebra extend uniquely to (homonymous) Σ*-algebra morphisms* $\rho : T_\Sigma(Var) \to \mathcal{T}$. These notions extend to algebraic specifications. Many common mathematical structures have been defined as Σ-algebras: boolean algebras, natural/integer/rational numbers, lists, sets, bags (or multisets), maps (e.g., for states, heaps), trees, queues, stacks, etc. [4, 25]. Here we only need maps, to represent program states, written with an infix "$\mapsto$" operation symbol for map entries (binary operation taking a program variable and an integer) and an associative and commutative comma "," symbol to separate them.

Let us fix the following: (1) an algebraic signature Σ, specifying some desired configuration syntax, with a distinguished sort *Cfg*, (2) a sort-wise infinite set of variables *Var*, and (3) a Σ-algebra $\mathcal{T}$, the *configuration model*, which may but need not be the initial Σ-algebra. The elements of $\mathcal{T}$ of sort *Cfg* are called *configurations*.

Definition 1. *[35] A matching logic formula, or a* ***pattern****, is a first-order logic (FOL) formula which allows* terms *in $T_{\Sigma,Cfg}(Var)$, called* ***basic patterns****, as* predicates. *We define the satisfaction $(\gamma,\rho) \models \varphi$ over configurations $\gamma \in \mathcal{T}_{Cfg}$, valuations $\rho : Var \to \mathcal{T}$ and patterns φ as follows (among the FOL constructs, we only show $\exists$):*

$$(\gamma,\rho) \models \exists X\ \varphi \ \ \textit{iff}\ \ (\gamma,\rho') \models \varphi \textit{ for some } \rho' : Var \to \mathcal{T} \textit{ with } \rho'(y) = \rho(y) \textit{ for all } y \in Var \backslash X$$

$$(\gamma,\rho) \models \pi \quad \textit{iff} \ \ \gamma = \rho(\pi) \ \ , \textit{ where } \pi \in T_{\Sigma,Cfg}(Var)$$

A basic pattern π is satisfied by all the configurations γ that *match* it; the ρ in $(\gamma,\rho) \models \pi$ can be thought of as the "witness" of the matching, and can be further constrained in a pattern. In the case of IMP, whose configurations have the form $\langle \texttt{code}, \sigma \rangle$, the pattern $\exists s\ (\langle\ \texttt{SUM},\ (\texttt{s} \mapsto s, \texttt{n} \mapsto n)\ \rangle\ \wedge\ n\ \geq_{Int}\ 0)$ matches the configurations with code `SUM` (from Section 2) and state binding program variables `s` and `n` respectively to integers s and $n \geq_{Int} 0$. We typically use `typewriter` for program variables and *italic* for mathematical variables in *Var*. Similarly, pattern $\langle \texttt{skip},\ (\texttt{s} \mapsto n *_{Int} (n +_{Int} 1)/_{Int} 2,\ \texttt{n} \mapsto 0)\rangle$ matches (with the same witness ρ, i.e., the same n) all the final configurations reachable (in IMP's transition system corresponding to its small-step semantics) from the configurations specified by the previous pattern (Section 5 shows how to formally prove it).

3.2 Abstract Matching Logic

Based on intuitions from first-order matching logic, we propose *abstract matching logic*, an abstract formalism for configuration properties. The abstraction consists of specifying properties of patterns and matching, rather fixing a concrete definition.

Definition 2. *An* ***abstract matching logic****, called simply a* ***matching logic*** *from here on, is a tuple $\mathcal{L} = (\Sigma, Var, \mathcal{T}, \mathcal{P}, \models)$ where:*

- *Σ is a multisorted signature containing a distinguished sort Cfg;*
- *Var is a sortwise infinite set of variables over the sorts of Σ.*
- *$\mathcal{T}$ is a Σ-algebra; its elements in $\mathcal{T}_{Cfg}$ are called* ***configurations****;*
- *$\mathcal{P}$ is a set of* ***patterns*** *φ, closed under the logical constructs $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \to \varphi_2$, and $\exists x\, \varphi$, where $x \in Var$;*
- *$\models\ \subseteq (\mathcal{T}_{Cfg} \times [Var \to \mathcal{T}]) \times \mathcal{P}$ is a* ***pattern matching*** *relation, written $(\gamma,\rho) \models \varphi$ and read "configuration $\gamma \in \mathcal{T}_{Cfg}$ matches pattern $\varphi \in \mathcal{P}$ with valuation $\rho : Var \to \mathcal{T}$ as witness". The pattern matching relation satisfies the following constraints: $(\gamma,\rho) \models \exists x\, \varphi$ iff there exists a ρ' such that $\rho'(y) = \rho(y)$ for all $y \in Var \setminus \{x\}$ and such that $(\gamma,\rho') \models \varphi$; and $(\gamma,\rho) \models \varphi_1 \wedge \varphi_2$ iff $(\gamma,\rho) \models \varphi_1$ and $(\gamma,\rho) \models \varphi_2$, and similarly for $\vee$ and $\to$.*

We also assume a "free-variable" function $FV : \mathcal{P} \to 2^{Var}$ on patterns with the expected properties: $FV(\exists x\, \varphi) = FV(\varphi) \setminus \{x\}$ and $FV(\varphi_1 \wedge \varphi_2) = FV(\varphi_1) \cup FV(\varphi_2)$, and similarly for $\vee$ and $\to$. Also, if ρ and ρ' agree on $FV(\varphi)$, then for all γ, $(\gamma,\rho) \models \varphi$ iff $(\gamma,\rho') \models \varphi$.

Thus, a matching logic essentially consists of a pattern matching relation that obeys some reasonable constraints. Note that the precise syntax and semantics of patterns are purposely left open: all we require is that certain reasonable syntactic constructs are available and have their expected compositional semantics. This allows us to instantiate our framework and the subsequent results not only to first-order, but also to higher-order, separation, or other logics.

Definition 3. *Pattern φ is* ***valid****, written $\models \varphi$, iff $(\gamma,\rho) \models \varphi$ for all $\gamma \in \mathcal{T}_{Cfg}$ and $\rho : Var \to \mathcal{T}$. Pattern φ is* ***structureless*** *iff for all $\rho : Var \to \mathcal{T}$ and $\gamma, \gamma' \in \mathcal{T}_{Cfg}$, $(\gamma,\rho) \models \varphi$ iff $(\gamma',\rho) \models \varphi$. Pattern φ is* ***weakly well-defined*** *(resp.* ***well-defined****) iff for any* *valuation $\rho : Var \to \mathcal{T}$ there exists at least one (resp. precisely one) configuration $\gamma \in \mathcal{T}_{Cfg}$ such that $(\gamma, \rho) \models \varphi$.*

The intuition is that a valid pattern contains no structural and no logical constraints, while a structureless pattern contains only logical but no structural constraints. Well-defined patterns have enough structural constraints to uniquely identify the matching configuration. In first-order matching logic all basic patterns π are well-defined, while patterns of the form $\pi_1 \vee \pi_2$ are weakly well-defined.

4. Reachability Rules

Unconditional reachability rules over first-order matching logic patterns were introduced in [33], which showed they can express certain operational semantics that do not require rule premises, such as reduction semantics with evaluation contexts [38]. They were also studied in [32], which showed they can express the Hoare triples of axiomatic semantics. In this section we introduce *conditional* reachability rules, a generalization which can naturally deal with any rewrite-based operational semantics.

Unless otherwise specified, from here on in this paper we assume an arbitrary but fixed matching logic $\mathcal{L} = (\Sigma, Var, \mathcal{T}, \mathcal{P}, \models)$ and give our definitions and proofs for this general framework, including the soudness of our proof system in Section 5.

Definition 4. *A **(conditional) reachability rule** is a sentence*

$$\varphi \Rightarrow \varphi' \ \text{ if } \ \varphi_1 \Rightarrow \varphi'_1 \bigwedge \ldots \bigwedge \varphi_n \Rightarrow \varphi'_n$$

*where $n \geq 0$ and where φ, φ', φ_1, φ'_1, ..., φ_n, φ'_n are matching logic patterns. We call φ the **left-hand side (LHS)** and φ' the **right-hand side (RHS)** of the rule. A rule is **unconditional** when $n = 0$, and is written $\varphi \Rightarrow \varphi'$. A **reachability system** is a set of reachability rules.*

4.1 Operational Semantics using Reachability Rules

As discussed in Section 2, in this paper we assume that operational semantics can be defined with particular rewrite rules of the form

$$cfg \Rightarrow cfg' \ \text{ if } \ b \bigwedge cfg_1 \Rightarrow cfg'_1 \bigwedge b_1 \bigwedge \ldots \bigwedge cfg_n \Rightarrow cfg'_n \bigwedge b_n,$$

which can be now seen as sugar for particular reachability rules

$$cfg \wedge b \wedge b_1 \wedge \ldots \wedge b_n \Rightarrow cfg' \ \text{ if } \ cfg_1 \Rightarrow cfg'_1 \bigwedge \ldots \bigwedge cfg_n \Rightarrow cfg'_n$$

Here the Boolean side conditions have been all conjuncted with the LHS pattern. Recall from Definition 2 that matching logic includes configuration terms as patterns and allows the use of FOL constructs, in particular conjunction, to build new patterns, so the above is a correct reachability rule, where φ is $cfg \wedge b \wedge b_1 \wedge \ldots \wedge b_n$, etc. For example, the rule $\mathbf{cond}_1$ in the big-step semantics of IMP in Figure 1 is syntactic sugar for the reachability rule

$$\langle \texttt{if}\, e\ s_1\ s_2,\ \sigma\rangle \wedge i \neq 0 \Rightarrow \langle\sigma'\rangle \ \text{ if } \ \langle e,\ \sigma\rangle \Rightarrow \langle i\rangle \bigwedge \langle s_1,\ \sigma\rangle \Rightarrow \langle\sigma'\rangle$$

From here on we assume that a language/calculus/system is semantically defined as a reachability system and, unless otherwise specified, fix an arbitrary reachability system $\mathcal{S}$ (with patterns in $\mathcal{L}$). It is irrelevant for the subsequent developments whether such rules represent a small-step, a big-step, or any other particular kind of operational semantics.

An operational semantics typically describes program behaviors by generating a transition system over program configurations, which can be used to associate a behavior to any given program in any given state. In some cases, e.g., small-step semantics, the transition system comprises all the atomic computational steps; in other cases, e.g., big-step semantics, the transition system consists of a binary relationship mapping configurations holding (fragments of) programs to their resulting configurations after evaluation.

We next show how $\mathcal{S}$ yields a transition system $(\mathcal{T}_{Cfg}, \to_{\mathcal{S}})$ over the configurations of $\mathcal{T}$.

Definition 5. *We inductively construct the relations $\mathcal{R}_k \subseteq \mathcal{T}_{Cfg} \times \mathcal{T}_{Cfg}$ for all $k \in \mathbb{N}$ as follows:*

- $\mathcal{R}_0 = \emptyset$
- $\mathcal{R}_{k+1} = \{\, (\gamma,\ \gamma') \ \mid$ *there exists some reachability rule*

$$\varphi \Rightarrow \varphi' \ \text{ if } \ \varphi_1 \Rightarrow \varphi'_1 \bigwedge \ldots \bigwedge \varphi_n \Rightarrow \varphi'_n$$

  in $\mathcal{S}$ and some valuation $\rho : Var \to \mathcal{T}$ such that:
  1. *$(\gamma, \rho) \models \varphi$ and $(\gamma', \rho) \models \varphi'$; and*
  2. *for all $\gamma_1, \ldots \gamma_n \in \mathcal{T}_{Cfg}$ with $(\gamma_i\ , \rho) \models \varphi_i$ for all $1 \leq i \leq n$ there exist $\gamma'_1, \ldots, \gamma'_n$ with $(\gamma'_i\ , \rho) \models \varphi'_i$ for all $1 \leq i \leq n$ such that $(\gamma_i,\ \gamma'_i) \in \mathcal{R}^*_k$, the transitive and reflexive closure of $\mathcal{R}_k$*

  $\}$

*Then $\to_{\mathcal{S}} \overset{def}{=} \bigcup_{k \geq 0} \mathcal{R}_k$, written infix, is the **transition relation induced by $\mathcal{S}$**, and $(\mathcal{T}_{Cfg}, \to_{\mathcal{S}})$ is the **transition system induced by $\mathcal{S}$**.*

Intuitively, $\mathcal{R}_1$ is the transition relation generated by using only unconditional rules or degenerate conditional rules whose conditions $\varphi_i \Rightarrow \varphi'_i$ specify pairs of identical configurations (because $\mathcal{R}^*_0 = \{(\gamma, \gamma) \mid \gamma \in \mathcal{T}_{Cfg}\}$) which therefore need no other reachability rule to be solved, $\mathcal{R}_2$ is generated using unconditional or conditional rules, but solving the conditions only with unconditional or degenerate conditional rules, and so on. In general, $\mathcal{R}_k$ is obtained by applying at most $k-1$ "nested" conditional rules. It is easy to see that $\mathcal{R}_k \subseteq \mathcal{R}_{k+1}$. In fact, the above explicitly defines the transition relation as the least fixed point relation $\to_{\mathcal{S}}$ that is compatible with all the rules in $\mathcal{S}$, where all rule conditions are interpreted as $\to_{\mathcal{S}}$-reachability.

If $\mathcal{L}$ is a first-order matching logic and $\mathcal{S}$ contains only rewrite rules, that is, rules whose patterns are all basic, then all the configurations γ, γ', γ_1, γ'_1, $\ldots$, γ_n, γ'_n in Definition 5 are uniquely determined by ρ, since $(\gamma, \rho) \models \pi$ iff $\gamma = \rho(\pi)$ for any basic pattern π (by Definition 1). In this case, $\to_{\mathcal{S}}$ becomes the usual transition relation induced by a (top-most) term rewrite system on a Σ-algebra. More specifically, if $\mathcal{S}$ is IMP's small-step semantics in Figure 1, then the following are valid transitions (SUM is the sum program in Section 2 and LOOP is its loop; for notational simplicity, we make no distinction between ground terms and their interpretation in $\mathcal{T}$):

$$\begin{array}{r}
\langle \texttt{SUM},\ (\texttt{s}\mapsto 7, \texttt{n}\mapsto 10)\rangle \ \to_{\mathcal{S}} \ \langle \texttt{LOOP},\ (\texttt{s}\mapsto 0, \texttt{n}\mapsto 10)\rangle \ \to_{\mathcal{S}} \\
\langle \texttt{if (n>0) (s:=s+n;n:=n-1;LOOP) skip},\ (\texttt{s}\mapsto 0, \texttt{n}\mapsto 10)\rangle \ \to_{\mathcal{S}} \\
\langle \texttt{if(10>0) (s:=s+n;n:=n-1;LOOP) skip},\ (\texttt{s}\mapsto 0, \texttt{n}\mapsto 10)\rangle \ \to_{\mathcal{S}} \\
\ldots \ \to_{\mathcal{S}} \ \langle \texttt{LOOP},\ (\texttt{s}\mapsto 10, \texttt{n}\mapsto 9)\rangle \ \to_{\mathcal{S}} \ \ldots \ \to_{\mathcal{S}} \\
\langle \texttt{LOOP},\ (\texttt{s}\mapsto 55, \texttt{n}\mapsto 0)\rangle \ \to_{\mathcal{S}} \ \ldots \ \to_{\mathcal{S}} \ \langle \texttt{skip},\ (\texttt{s}\mapsto 55, \texttt{n}\mapsto 0)\rangle
\end{array}$$

In computing the transitions above, we need to go up to 3 nested conditional rules in Definition 5, i.e., $k = 4$. On the other hand, if $\mathcal{S}$ is the IMP's big-step semantics in Figure 1, then we have

$$\langle \texttt{SUM},\ (\texttt{s}\mapsto 7, \texttt{n}\mapsto 10)\rangle \ \to_{\mathcal{S}} \ \langle \texttt{s}\mapsto 55, \texttt{n}\mapsto 0\rangle$$

in one transition step, but in order to compute that we need to apply more than 40 nested conditional rules.

We next define a strong notion of validity, based entirely on the transition relation above. Definition 10 introduces our actual notion of validity, partial correctness in the sense of Hoare-style validity, which also takes into account termination; as seen in Definition 7, defining termination requires strong validity.

Definition 6. *Given a valuation $\rho : Var \to \mathcal{T}$, an unconditional reachability rule $\varphi \Rightarrow \varphi'$ is **ρ-strongly-valid**, written $\mathcal{S}, \rho \overset{s}{\models} \varphi \Rightarrow \varphi'$, iff for any $\gamma \in \mathcal{T}_{Cfg}$ with $(\gamma, \rho) \models \varphi$ there is some $\gamma' \in \mathcal{T}_{Cfg}$ such that $(\gamma', \rho) \models \varphi'$ and $\gamma \to^{\star}_{\mathcal{S}} \gamma'$. Rule $\varphi \Rightarrow \varphi'$ is **strongly valid**, written $\mathcal{S} \overset{s}{\models} \varphi \Rightarrow \varphi'$, iff it is ρ-strongly-valid for each $\rho : Var \to \mathcal{T}$.*

In the case of IMP, if $\rho(\sigma) = (\texttt{s}\mapsto 7, \texttt{n}\mapsto 10)$ and $\rho(\sigma') = (\texttt{s}\mapsto 55, \texttt{n}\mapsto 0)$ then, as seen above, $\mathcal{S}, \rho \overset{s}{\models} \langle \texttt{SUM},\ \sigma\rangle \Rightarrow \langle \texttt{skip},\ \sigma'\rangle$ when $\mathcal{S}$ is the small-step semantics of IMP, and $\mathcal{S}, \rho \overset{s}{\models} \langle \texttt{SUM},\ \sigma\rangle \Rightarrow \langle\sigma'\rangle$ when $\mathcal{S}$ is the big-step semantics of IMP in Figure 1, respectively. More interestingly, we can show that $\mathcal{S} \overset{s}{\models} \langle \texttt{SUM},\ (\texttt{s}\mapsto s, \texttt{n}\mapsto n)\rangle \wedge n \geq_{Int} 0 \Rightarrow \langle \texttt{skip},\ (\texttt{s}\mapsto n *_{Int} (n +_{Int} 1)/_{Int} 2, \texttt{n}\mapsto 0)\rangle$ with the small-step semantics, and a similar property with the big-step semantics, but the proof would be tedious and low level at this moment, as it would need to directly involve the transition system $(\mathcal{T}_{Cfg}, \rightarrow_{\mathcal{S}})$. Section 5 shows how such properties can be derived more abstractly, using the reachability logic proof system.

An operational semantics should also formally say when a program terminates, or when it diverges. In some cases, e.g., small-step semantics, nontermination is captured by the existence of an infinite number of transitions starting with the given configuration; in other cases, e.g., big-step semantics, nontermination is captured by the inability to perform a step in the transition system due to an infinite sequence of nested attempts to fulfill the rule premises.

We next define termination of configurations with respect to $\mathcal{S}$, capturing both cases above. Our definition is based on a partial order on configurations which is inspired from quasi-decreasing orders for conditional term rewriting systems [13]. Our definition is also somewhat related to operational termination of conditional term rewrite systems [20], although the latter is a property of a rewrite theory while our notion of termination refers to a particular configuration in a particular model.

Definition 7. *Let $(\mathcal{T}_{Cfg}, \succ)$ be the **termination dependence** relation:*

- *$\gamma \succ \gamma'$ if $\gamma \rightarrow_S \gamma'$; and*
- *$\gamma \succ \gamma'$ if there exists a rule $\varphi \Rightarrow \varphi'$ if $\varphi_1 \Rightarrow \varphi'_1 \bigwedge \ldots \bigwedge \varphi_n \Rightarrow \varphi'_n$ in $\mathcal{S}$, valuation $\rho : Var \rightarrow \mathcal{T}$, and index $1 \leq i \leq n$ such that:*
  1. *$(\gamma, \rho) \models \varphi$;*
  2. *$\mathcal{S}, \rho \overset{*}{\models} \varphi_j \Rightarrow \varphi'_j$ for each $1 \leq j < i$; and*
  3. *$(\gamma', \rho) \models \varphi_i$.*

*Then $\gamma \in \mathcal{T}_{Cfg}$ **terminates** iff there are no infinite decreasing $\succ$ chains starting at γ, and γ **diverges** otherwise. We also let $\succeq$ denote the partial order associated to $\succ$, i.e., its reflexive and transitive closure.*

Our definition of termination above mimics the application of conditional rules in the configuration model, in that conditions are solved in order and a condition is considered only if all the previous conditions are successfully solved.

Let us consider our particular IMP language again. In Section 2 we informally claimed that `SUM` always terminates, `SUM'` only terminates when `n` $\leq$ `0`, and $\texttt{SUM}_\infty$ never terminates. We can now make these claims formal. For `SUM`, we can show that any configuration γ of the form $\langle \texttt{SUM}, \sigma \rangle$ terminates with any of the two semantics in Figure 1, for any state σ (including σ's which lack `s` or `n`). For `SUM'`, any configuration of the form $\langle \texttt{SUM'}, (\texttt{n} \mapsto n, \sigma) \rangle$ with $n \leq 0$ terminates in both semantics, whether or not σ binds `s`. However, note that our informal claim in Section 2 that "`SUM'` only terminates when `n` $\leq$ `0`" was (purposely) imprecise. Indeed, configurations $\langle \texttt{SUM'}, \sigma \rangle$ with `n` or `s` undefined in σ also terminate. Finally, our informal claim that "$\texttt{SUM}_\infty$ never terminates" was also imprecise for similar reasons. Stated precisely, configurations of the form $\langle \texttt{SUM}, (\texttt{n} \mapsto n, \texttt{s} \mapsto s, \sigma) \rangle$ diverge. It is interesting to note that such configurations diverge for different reasons in the two semantics, descending by the first bullet of Definition 7 in small-step semantics, and by the second bullet in big-step semantics.

Definition 8. *A pattern φ **terminates** (resp. **diverges**), written $\mathcal{S} \models \varphi\downarrow$ (resp. $\mathcal{S} \models \varphi\uparrow$), iff for all $\gamma \in \mathcal{T}_{Cfg}$ and for all $\rho : Var \rightarrow \mathcal{T}$, if $(\gamma, \rho) \models \varphi$ then γ terminates (resp. diverges).*

In the case of IMP with $\mathcal{S}$ either its small-step or its big-step semantics, from the discussion above we can conclude

$$\mathcal{S} \models \langle \texttt{SUM}, \sigma \rangle\downarrow$$
$$\mathcal{S} \models (\langle \texttt{SUM'}, (\texttt{n} \mapsto n, \sigma) \rangle \wedge n \leq_{Int} 0 \vee \langle \texttt{SUM'}, \sigma \rangle \wedge (\texttt{n} \notin Dom(\sigma) \vee \texttt{s} \notin Dom(\sigma))\downarrow$$
$$\mathcal{S} \models (\langle \texttt{SUM'}, (\texttt{n} \mapsto n, \texttt{s} \mapsto s, \sigma) \rangle \wedge n >_{Int} 0)\uparrow$$
$$\mathcal{S} \models \langle \texttt{SUM}_\infty, (\texttt{n} \mapsto n, \texttt{s} \mapsto s, \sigma) \rangle\uparrow$$

Section 5 shows how to prove divergence using our proof system for reachability. Proving termination is left for future work.

Recall that $\mathcal{S}$ is an arbitrary reachability system, thought of as a "semantics". However, not all reachability systems are meaningful as semantics in all situations. Consider a reachability system containing a rule of the form $\varphi \Rightarrow \mathit{false}$. Such a rule is not just semantically useles (because it generates no transitions), but also makes reachability reasoning unsound, because even $\mathcal{S}$'s own rules are not all valid ($\mathcal{S} \overset{*}{\models} \varphi \Rightarrow \mathit{false}$ does not hold, though we expect $\mathcal{S} \overset{*}{\models} \mu$ for any unconditional $\mu \in \mathcal{S}$). It is therefore not surprising that some of the subsequent results need to impose additional constraints on the rules of $\mathcal{S}$, such as the following:

Definition 9. *Rule $\varphi \Rightarrow \varphi'$ if $\varphi_1 \Rightarrow \varphi'_1 \bigwedge \ldots \bigwedge \varphi_n \Rightarrow \varphi'_n$ is **(weakly) well-defined** iff φ', φ_1, ..., φ_n are (weakly) well-defined. Reachability system $\mathcal{S}$ is **(weakly) well-defined** iff all its rules are.*

Since operational semantics rules contain only configuration terms except possibly for their LHS patterns (see discussion at beginning of Section 4.1), and since configuration terms are basic patterns, which are always well-defined, we believe that it is safe to say that all reachability systems of interest are expected to be well-defined. Nevertheless, weak well-definedness suffices for the soundness of reachability logic, although we need full well-definedness for completeness.

4.2 Specifying Program Properties using Reachability Rules

Reachability rules can specify not only operational semantics, but also program properties. In fact, each Hoare triple can be translated into a particular reachability rule [32], although the translation needs to be mechanized separately for each language. However, it is not recommend to follow this route when specifying program properties, because Hoare triples can be more complex than reachability rules expressing the same property, even without the additional complexity added by the mechanical translation. Consider, for example, the following Hoare triple expressing `SUM`'s property:

$$\{\texttt{n} = \texttt{oldn} \wedge \texttt{n} \geq 0\} \;\; \texttt{SUM} \;\; \{\texttt{s} = \texttt{oldn} * (\texttt{oldn} + 1) \,/\, 2 \wedge \texttt{n} = 0\}$$

The introduction of the additional `oldn` variable follows a common Hoare logic "trick" to save the initial value of `n`. Following [32], the above Hoare triple translates mechanically into the reachability rule

$$\exists s, n\, (\langle \texttt{SUM},\ (\texttt{s} \mapsto s, \texttt{n} \mapsto n) \rangle \wedge n = oldn \wedge n \geq_{Int} 0) \;\; \Rightarrow$$
$$\exists s, n\, (\langle \texttt{skip},\ (\texttt{s} \mapsto s, \texttt{n} \mapsto n) \rangle \wedge s = oldn *_{Int} (oldn +_{Int} 1) /_{Int} 2 \wedge n = 0)$$

On the other hand, with the configurations of IMP's big-step semantics in Figure 1, we can express the same property as follows:

$$\langle \texttt{SUM}, (\texttt{s} \mapsto s, \texttt{n} \mapsto n) \rangle \wedge n \geq_{Int} 0 \;\Rightarrow\; \langle (\texttt{s} \mapsto n *_{Int} (n +_{Int} 1) /_{Int} 2, \texttt{n} \mapsto 0) \rangle$$

In words, if we execute the configuration holding the program `SUM` and a state binding program variables `s` and `n` to integers s and $n \geq 0$ using IMP's big-step semantics, then we reach a configuration holding a state that binds program variables `s` and `n` to the sum of numbers up to n and to 0, respectively. Technically, s and n are variables of sort *Int*; one can also think of them as "symbolic" integers. On the other hand, `s` and `n` are constants of sort *PVar*.

One could argue that the Hoare triple above is more natural because it is more compact and the FOL specifications make direct use of program's variables. However, one should note that the reachability rule is more informative, since it also states that `s` and `n` must be available in the state before `SUM` is executed. To state these properties using Hoare logic we need additional specification contents, e.g. definedness predicates. Also, Hoare logic conflating program variables (like `s`, `n`) and specification variables (like `oldn`) is often a source of complexity and confusion, particularly in combination with substitution and pointers. Unlike Hoare triples, which only specify properties about final program states, reachability rules can also specify properties of intermediate states as reachability rules where the right hand side has some intermediate code. Hoare triples correspond to reachability rules whose RHS holds the empty code, like the one above. We refer the reader to [32] for more details on the expressiveness of unconditional reachability rules.

4.3 Validity and ω-Closure

In Hoare logic, a triple $\{pre\}$ `code` $\{post\}$ is (semantically) valid, in the sense of partial correctness, iff for any state that satisfies *pre*, if `code` terminates then the resulting state satisfies *post*. This elegant definition has the luxury of relying on another formal semantics of the target language which provides the language-specific notions of "state", "satisfaction", and "termination". Since here everything happens in a single language-independent framework, we generalize the notion of validity as follows:

Definition 10. *Given a valuation $\rho : Var \to \mathcal{T}$, an unconditional reachability rule $\varphi \Rightarrow \varphi'$ is ρ-**valid**, written $\mathcal{S}, \rho \models \varphi \Rightarrow \varphi'$, iff for any $\gamma \in \mathcal{T}_{Cfg}$ with $(\gamma, \rho) \models \varphi$, if γ terminates then there is some $\gamma' \in \mathcal{T}_{Cfg}$ such that $(\gamma', \rho) \models \varphi'$ and $\gamma \rightarrow^{\star}_{\mathcal{S}} \gamma'$. Rule $\varphi \Rightarrow \varphi'$ is **valid**, written $\mathcal{S} \models \varphi \Rightarrow \varphi'$, iff it is ρ-valid for each $\rho : Var \to \mathcal{T}$.*

The major difference between our validity and Hoare validity is that the language-specific notions of "state" and "code" have been replaced by the language-independent notion of "configurations". It is not hard to see that in the case of IMP, with $\mathcal{S}$ either of the semantics in Figure 1, this notion of validity becomes the usual Hoare logic validity when the reachability rule $\varphi \Rightarrow \varphi'$ corresponds to a Hoare triple as shown in Section 4.2.

If the rule LHS terminates, strong validity and validity coincide:

Proposition 1. *The following hold:*

1. *If $\mathcal{S} \models^{s} \varphi \Rightarrow \varphi'$ then $\mathcal{S} \models \varphi \Rightarrow \varphi'$;*
2. *If $\mathcal{S} \models \varphi\!\downarrow$ then $\mathcal{S} \models^{s} \varphi \Rightarrow \varphi'$ iff $\mathcal{S} \models \varphi \Rightarrow \varphi'$.*

In Hoare logic divergence can be indirectly specified using Hoare triples with postcondition *false*. We can similarly reduce proving divergence to proving a reachability rule whose RHS is *false*, provided our arbitrary matching logic $\mathcal{L}$ has a pattern *false* matched by no configurations (in first-order matching logic we have the FOL *false* formula): $\mathcal{S} \models \varphi\!\uparrow$ iff $\mathcal{S} \models \varphi \Rightarrow false$. Therefore, any complete proof system for reachability can also prove divergence. It turns out, however, that it is necessary (for the completeness theorem) and convenient to refer to divergence directly.

Definition 11. *We let $\mathcal{S}^{\omega}$, called the ω-**closure** of $\mathcal{S}$, be the reachability system extending $\mathcal{S}$ as follows:*

- *Add to Σ a new constant ω of sort Cfg;*
- *Add to $\mathcal{T}_{Cfg}$ a new element $\mathcal{T}_{\omega}$;*
- *Add to $\mathcal{S}$ a new rule, $\omega \Rightarrow \omega$;*
- *For each rule $\varphi \Rightarrow \varphi'$ if $\varphi_1 \Rightarrow \varphi'_1 \bigwedge \ldots \bigwedge \varphi_n \Rightarrow \varphi'_n$ in $\mathcal{S}$ and each $1 \le i \le n$, add to $\mathcal{S}$ a conditional reachability rule*

$$\varphi \Rightarrow \omega \;\text{ if }\; \varphi_1 \Rightarrow \varphi'_1 \bigwedge \ldots \bigwedge \varphi_{i-1} \Rightarrow \varphi'_{i-1} \bigwedge \varphi_i \Rightarrow \omega.$$

*By convention $(\mathcal{S}^{\omega})^{\omega} = \mathcal{S}^{\omega}$ and we call $\mathcal{S}$ ω-**closed** iff $\mathcal{S} = \mathcal{S}^{\omega}$.*

The ω -closure operation is algorithmic and easy to implement. Since $\mathcal{T}_{\omega}$ is the only configuration that matches ω, we conclude that ω is well-defined. In fact, the ω-closure operation does not affect well-definedness: $\mathcal{S}$ is (weakly) well-defined iff $\mathcal{S}^{\omega}$ is (weakly) well-defined. Moreover, the additional rules are semantically irrelevant:

Proposition 2. *The following equivalences hold for all configurations $\gamma, \gamma' \in \mathcal{T}_{Cfg}$ and for all patterns $\varphi, \varphi' \in \mathcal{P}$:*

- *$\gamma \rightarrow_{\mathcal{S}} \gamma'$ iff $\gamma \rightarrow_{\mathcal{S}^{\omega}} \gamma'$;*
- *γ terminates for $\mathcal{S}$ iff γ terminates for $\mathcal{S}^{\omega}$;*
- *$\mathcal{S} \models \varphi \Rightarrow \varphi'$ iff $\mathcal{S}^{\omega} \models \varphi \Rightarrow \varphi'$.*

Therefore, the ω-closure has no semantic effect. It only has proof-theoretical merit, ensuring we can prove divergence as follows:

$$\varphi \Rightarrow \varphi' \;\text{ if }\; \varphi_1 \Rightarrow \varphi'_1 \bigwedge \cdots \bigwedge \varphi_n \Rightarrow \varphi'_n \;\in\; \mathcal{A}$$
$$\psi \text{ is a structureless pattern}$$

$$\textbf{Axiom}:\quad \frac{\mathcal{A} \cup C \vdash \varphi_1 \wedge \psi \Rightarrow \varphi'_1 \quad \cdots \quad \mathcal{A} \cup C \vdash \varphi_n \wedge \psi \Rightarrow \varphi'_n}{\mathcal{A} \vdash_C \varphi \wedge \psi \Rightarrow \varphi' \wedge \psi}$$

$$\textbf{Reflexivity}:\quad \mathcal{A} \vdash \varphi \Rightarrow \varphi$$

$$\textbf{Transitivity}:\quad \frac{\mathcal{A} \vdash_C \varphi_1 \Rightarrow \varphi_2 \qquad \mathcal{A} \cup C \vdash \varphi_2 \Rightarrow \varphi_3}{\mathcal{A} \vdash_C \varphi_1 \Rightarrow \varphi_3}$$

$$\textbf{Consequence}:\quad \frac{\models \varphi_1 \to \varphi'_1 \qquad \mathcal{A} \vdash_C \varphi'_1 \Rightarrow \varphi'_2 \qquad \models \varphi'_2 \to \varphi_2}{\mathcal{A} \vdash_C \varphi_1 \Rightarrow \varphi_2}$$

$$\textbf{Case Analysis}:\quad \frac{\mathcal{A} \vdash_C \varphi_1 \Rightarrow \varphi \qquad \mathcal{A} \vdash_C \varphi_2 \Rightarrow \varphi}{\mathcal{A} \vdash_C \varphi_1 \vee \varphi_2 \Rightarrow \varphi}$$

$$\textbf{Abstraction}:\quad \frac{\mathcal{A} \vdash_C \varphi \Rightarrow \varphi' \qquad \text{where } X \cap FV(\varphi') = \emptyset}{\mathcal{A} \vdash_C \exists X\, \varphi \Rightarrow \varphi'}$$

$$\textbf{Circularity}:\quad \frac{\mathcal{A} \vdash_{C \cup \{\varphi \Rightarrow \varphi'\}} \varphi \Rightarrow \varphi'}{\mathcal{A} \vdash_C \varphi \Rightarrow \varphi'}$$

Figure 3. Reachability logic proof system.

Proposition 3. *If $\mathcal{S}$ is ω-closed, then $\mathcal{S} \models \varphi\!\uparrow$ iff $\mathcal{S} \models \varphi \Rightarrow \omega$.*

5. Proof System and Soundness

Figure 3 shows the reachability logic proof system. The target language is given as a weakly well-defined reachability system $\mathcal{S}$. The soundness result (Theorem 1) guarantees that $\mathcal{S} \models \varphi \Rightarrow \varphi'$ if $\mathcal{S} \vdash \varphi \Rightarrow \varphi'$ is derivable. Note that the proof system derives more general sequents of the form $\mathcal{A} \vdash_C \varphi \Rightarrow \varphi'$, where $\mathcal{A}$ and C are sets of reachability rules. The rules in $\mathcal{A}$ are called *axioms* and rules in C are called *circularities*. If C does not appear in a sequent, then it means it is empty: $\mathcal{A} \vdash \varphi \Rightarrow \varphi'$ is a shorthand for $\mathcal{A} \vdash_{\emptyset} \varphi \Rightarrow \varphi'$. Initially, C is empty and $\mathcal{A}$ is $\mathcal{S}$. During the proof, circularities can be added to C via the Circularity rule and flushed into $\mathcal{A}$ by the Transitivity or Axiom rules.

The intuition is that the rules in $\mathcal{A}$ can be assumed valid, while the rules in C have been postulated but not yet justified. After making concrete progress it becomes (coinductively) valid to rely on them. The intuition for a sequent $\mathcal{A} \vdash_C \varphi \Rightarrow \varphi'$, read "$\mathcal{A}$ with circularities C proves $\varphi \Rightarrow \varphi'$", is that $\varphi \Rightarrow \varphi'$ is true if the rules in $\mathcal{A}$ are true and the rules in C are true after making progress, and that if C is nonempty then φ reaches φ' (or diverges) after at least one transition.

With this in mind, let us discuss the proof rules.

Axiom states that a trusted rule can be used in any logical context, or frame. The logical frame is formalized as a structureless pattern ψ, as it is meant to only add logical but no structural constraints. Incorporating framing into the axiom rule is necessary to make logical constraints available while proving the conditions of the axiom hold. Since reachability logic keeps a clear separation between program variables and logical variables the logical constraints are persistent, that is, they do not interfere with the dynamic nature of the operational rules and can therefore be safely used for framing. This is not the case for structural constraints. Consider, for example, a structural constraint given as pattern $\langle$`skip`, $\sigma\rangle$. We cannot use this pattern as a frame ψ for the rule **skip** of the big-step semantics of IMP, because $\langle$`skip`, $\sigma\rangle \wedge \langle\sigma\rangle$ is matched by no pattern, same as *false*, so the proof system would unsoundly derive $\langle$`skip`, $\sigma\rangle \Rightarrow false$. Additionally, note that the circularities are released as trusted axioms when deriving the rule's conditions, which is consistent with the intuition above for sequents.

General macros					
$\mathtt{SUM}$	$\equiv$	`s := 0; while (n>0) (s := s+n; n := n-1)`	$\mathtt{LOOP}$	$\equiv$	`while (n>0) (s := s+n; n := n-1)`
$\mathtt{S}_1$	$\equiv$	`s := s + n; n := n - 1; LOOP`	$\mathtt{S}_2$	$\equiv$	`n := n - 1; LOOP`
$\mathtt{IF}$	$\equiv$	`if (n > 0) then` $\mathtt{S}_1$ `else skip`	$\mathtt{sum}_{\mathtt{inv}}(n, n')$	$\equiv$	$(n -_{Int} n') *_{Int} (n +_{Int} n' +_{Int} 1)/_{Int}\, 2$
$\varphi_{\mathtt{SUM}}$	$\equiv$	$\langle \mathtt{SUM},\ (\mathtt{s} \mapsto s, \mathtt{n} \mapsto n)\rangle \wedge n \geq_{Int} 0$	$\varphi_{\mathtt{IF}}$	$\equiv$	$\langle \mathtt{IF},\ (\mathtt{s} \mapsto \mathtt{sum}_{\mathtt{inv}}(n, n'), \mathtt{n} \mapsto n')\rangle$
$\varphi_{\mathtt{INV}}$	$\equiv$	$\langle \mathtt{LOOP},\ (\mathtt{s} \mapsto \mathtt{sum}_{\mathtt{inv}}(n, n'), \mathtt{n} \mapsto n')\rangle \wedge n' \geq_{Int} 0$	$\varphi^{after}_{\mathtt{LOOP}}$	$\equiv$	$\langle \mathtt{LOOP},\ (\mathtt{s} \mapsto \mathtt{sum}_{\mathtt{inv}}(n, n' -_{Int} 1), \mathtt{n} \mapsto n' -_{Int} 1)\rangle$
$\varphi_{\mathtt{S}_1}$	$\equiv$	$\langle \mathtt{S}_1,\ (\mathtt{s} \mapsto \mathtt{sum}_{\mathtt{inv}}(n, n'), \mathtt{n} \mapsto n')\rangle$	$\varphi_{\mathtt{S}_2}$	$\equiv$	$\langle \mathtt{S}_2,\ (\mathtt{s} \mapsto \mathtt{sum}_{\mathtt{inv}}(n, n' -_{Int} 1), \mathtt{n} \mapsto n')\rangle$

Small-step macros			Big-step macros		
φ	$\equiv$	$\langle \mathtt{skip},\ (\mathtt{s} \mapsto n *_{Int} (n +_{Int} 1)/_{Int}\, 2, \mathtt{n} \mapsto 0)\rangle$	φ	$\equiv$	$\langle (\mathtt{s} \mapsto n *_{Int} (n +_{Int} 1)/_{Int}\, 2, \mathtt{n} \mapsto 0)\rangle$
μ	$\equiv$	$\exists n'\, \varphi_{\mathtt{INV}} \Rightarrow \varphi$	μ	$\equiv$	$\exists n'\, \varphi_{\mathtt{INV}} \Rightarrow \varphi$

Small-step proof derivation ($\mathcal{S}$ is IMP's small-step semantics)		Big-step proof derivation ($\mathcal{S}$ is IMP's big-step semantics)	
1. $\mathcal{S} \vdash \varphi_{\mathtt{SUM}} \Rightarrow \varphi_{\mathtt{INV}} \wedge n' =_{Int} n$	[**asgn**$_2$, **seq**$_1$, **seq**$_2$]	1. $\mathcal{S} \cup \{\mu\} \vdash \exists n'\, \varphi_{\mathtt{INV}} \Rightarrow \varphi$	[μ]
2. $\mathcal{S} \vdash \varphi_{\mathtt{INV}} \Rightarrow \varphi_{\mathtt{IF}} \wedge n' \geq_{Int} 0$	[**while**]	2. $\mathcal{S} \cup \{\mu\} \vdash \varphi^{after}_{\mathtt{LOOP}} \wedge n' >_{Int} 0 \Rightarrow \varphi$	[**Consequence**(1)]
3. $\mathcal{S} \vdash_{\{\mu\}} \varphi_{\mathtt{IF}} \wedge n' >_{Int} 0 \Rightarrow \varphi_{\mathtt{S}_1} \wedge n' >_{Int} 0$	[**lookup**, **>**$_1$, **>**$_3$, **cond**$_1$, **cond**$_2$]	3. $\mathcal{S} \vdash_{\{\mu\}} \varphi_{\mathtt{S}_2} \wedge n' >_{Int} 0 \Rightarrow \varphi$	[**lookup**, **int**, **-**, **asgn**, **seq**(2)]
4. $\mathcal{S} \vdash_{\{\mu\}} \varphi_{\mathtt{S}_1} \wedge n' >_{Int} 0 \Rightarrow \varphi_{\mathtt{S}_2} \wedge n' >_{Int} 0$	[**lookup**,**+**$_1$,**+**$_2$,**+**$_3$,**asgn**$_1$,**asgn**$_2$,**seq**$_1$,**seq**$_2$]	4. $\mathcal{S} \vdash_{\{\mu\}} \varphi_{\mathtt{S}_1} \wedge n' >_{Int} 0 \Rightarrow \varphi$	[**lookup**, **int**, **+**, **asgn**, **seq**(3)]
5. $\mathcal{S} \vdash_{\{\mu\}} \varphi_{\mathtt{S}_2} \wedge n' >_{Int} 0 \Rightarrow \varphi^{after}_{\mathtt{LOOP}} \wedge n' >_{Int} 0$	[**lookup**,**-**$_1$,**-**$_3$,**asgn**$_1$,**asgn**$_2$,**seq**$_1$,**seq**$_2$]	5. $\mathcal{S} \vdash_{\{\mu\}} \varphi_{\mathtt{INV}} \wedge n' >_{Int} 0 \Rightarrow \varphi$	[**lookup**, **int**, **>**, **while**$_2$(4)]
6. $\mathcal{S} \vdash_{\{\mu\}} \varphi_{\mathtt{S}_2} \wedge n' >_{Int} 0 \Rightarrow \exists n'\, \varphi_{\mathtt{INV}}$	[**Consequence**(5)]	6. $\mathcal{S} \vdash_{\{\mu\}} \varphi_{\mathtt{INV}} \wedge n' =_{Int} 0 \Rightarrow \varphi$	[**lookup**, **int**, **>**, **while**$_1$]
7. $\mathcal{S} \cup \{\mu\} \vdash \exists n'\, \varphi_{\mathtt{INV}} \Rightarrow \varphi$	[μ]	7. $\mathcal{S} \vdash_{\{\mu\}} \varphi_{\mathtt{INV}} \Rightarrow \varphi$	[**Case Analysis**(5, 6)]
8. $\mathcal{S} \vdash_{\{\mu\}} \varphi_{\mathtt{IF}} \wedge n' >_{Int} 0 \Rightarrow \varphi$	[**Transitivity**(3, 4, 6, 7)]	8. $\mathcal{S} \vdash_{\{\mu\}} \exists n'\, \varphi_{\mathtt{INV}} \Rightarrow \varphi$	[**Abstraction**(7)]
9. $\mathcal{S} \vdash_{\{\mu\}} \varphi_{\mathtt{IF}} \wedge n' =_{Int} 0 \Rightarrow \varphi$	[**lookup**, **>**$_1$, **>**$_3$, **cond**$_1$, **cond**$_3$]	9. $\mathcal{S} \vdash \exists n'\, \varphi_{\mathtt{INV}} \Rightarrow \varphi$	[**Circularity**(8)]
10. $\mathcal{S} \vdash_{\{\mu\}} \varphi_{\mathtt{IF}} \wedge n' \geq_{Int} 0 \Rightarrow \varphi$	[**Case Analysis**(8, 9)]	10. $\mathcal{S} \vdash \varphi_{\mathtt{INV}} \wedge n' =_{Int} n \Rightarrow \varphi$	[**Consequence**(9)]
11. $\mathcal{S} \vdash_{\{\mu\}} \exists n'\, \varphi_{\mathtt{INV}} \Rightarrow \varphi$	[**Transitivity**(2, 10); **Abstraction**]	11. $\mathcal{S} \vdash \varphi_{\mathtt{SUM}} \Rightarrow \varphi$	[**int**, **asgn**, **skip**, **seq**(10)]
12. $\mathcal{S} \vdash \exists n'\, \varphi_{\mathtt{INV}} \Rightarrow \varphi$	[**Circularity**(11)]		
13. $\mathcal{S} \vdash \varphi_{\mathtt{SUM}} \Rightarrow \exists n'\, \varphi_{\mathtt{INV}}$	[**Consequence**(1)]		
14. $\mathcal{S} \vdash \varphi_{\mathtt{SUM}} \Rightarrow \varphi$	[**Transitivity**(13, 12)]		

Figure 2. Formal reachability logic proofs for SUM. Simple Consequence rules used to perform domain reasoning are elided for readability.

Reflexivity and transitivity correspond to corresponding closure properties of the reachability relation. Reflexivity requires C to be empty to meet the requirement above, that a reachability property derived with nonemtpy C takes one or more steps. Transitivity releases the circularities as axioms for the second premise, because if there are any circularities to release the first premise is guaranteed to make progress.

Consequence and Case Analysis are adapted from Hoare logic. In Hoare logic Case Analysis is typically a derived rule, and we could probably derive it for particular languages, but there is no way to prove it language-independently. Ignoring circularities, we can think simplistically of these five rules as a rigorous infrastructure for symbolic execution.

Abstraction allows us to hide irrelevant details of φ behind an existential quantifier, which is particularly useful in combination with the next proof rule.

Circularity allows to make a new circularity claim at any moment during a proof. We typically make such claims for code with repetitive behaviors, such as loops, recursive functions, jumps, etc. If we succeed in proving the claim using itself as a circularity, then the claim holds. This circular reasoning would obviously be unsound if circularities were unrestricted, but requiring progress before circularities can be used ensures that only diverging executions can correspond to endless invocation of a circularity.

Figure 2 shows detailed formal proofs that the SUM program (Section 2) indeed calculates the sum of the first n natural numbers in s, for the small-step and big-step semantics of IMP from Figure 1. In the small-step case (left column) the circularity corresponding to the loop is used via the Transitivity rule, while in the big-step case (right column) the circularity is used via the Axiom rule. Below we discuss these proofs informally (LOOP is the while loop of SUM).

In the small-step case, the specification $\varphi_{\text{SUM}} \Rightarrow \varphi$ is

$$\langle \mathtt{SUM}, (\mathtt{s} \mapsto s, \mathtt{n} \mapsto n)\rangle \wedge n \geq_{Int} 0 \Rightarrow \langle \mathtt{skip}, (\mathtt{s} \mapsto n *_{Int} (n +_{Int} 1)/_{Int} 2, \mathtt{n} \mapsto 0)\rangle.$$

We begin by transitivity through $\exists n'\, \varphi_{\text{INV}}$, where φ_{INV} is the pattern

$$\langle \mathtt{LOOP},\ (\mathtt{s} \mapsto (n -_{Int} n') *_{Int} (n +_{Int} n' +_{Int} 1)/_{Int} 2,\ \mathtt{n} \mapsto n')\rangle \wedge n' \geq_{Int} 0.$$

$\varphi_{\text{SUM}} \Rightarrow \exists n'\, \varphi_{\text{INV}}$ holds by running the operational semantics on SUM until the pattern $\langle \mathtt{LOOP},\ (\mathtt{s} \mapsto 0, \mathtt{n} \mapsto n)\rangle \wedge n \geq_{Int} 0$ is reached, and abstracting this as $\exists n'\, \varphi_{\text{INV}}$ by Consequence. The property $\mu \equiv \exists n'\, \varphi_{\text{INV}} \Rightarrow \varphi$ is proved by Circularity. Abstraction removes the quantifier and fixes an arbitrary n', allowing us to unroll the loop into a conditional by the while rule. This progress releases the circularity. We continue by Case Analysis on $n' =_{Int} 0 \vee n' >_{Int} 0$, running the operational semantics in each case. When $n' =_{Int} 0$ the goal is reached directly, and when $n' >_{Int} 0$ we reach a configuration implying $\exists n'\, \varphi_{\text{INV}}$ and finish by applying the recently-added axiom.

In the big-step case the specification $\varphi_{\text{SUM}} \Rightarrow \varphi$ is now

$$\langle \mathtt{SUM},\ (\mathtt{s} \mapsto s, \mathtt{n} \mapsto n)\rangle \wedge n \geq_{Int} 0 \Rightarrow \langle (\mathtt{s} \mapsto n *_{Int} (n +_{Int} 1)/_{Int} 2, \mathtt{n} \mapsto 0)\rangle$$

As before, we prove $\mu \equiv \exists n'\, \varphi_{\text{INV}} \Rightarrow \varphi$, with the same φ_{INV} as before. We reach $\exists n'\, \varphi_{\text{INV}}$ from φ_{SUM} by applying the big-step semantics of assignment and sequential composition. The difference is that this is reached in a premise of applications of conditional axioms, rather than a premise of Transitivity. Property $\exists n'\, \varphi_{\text{INV}} \Rightarrow \varphi$ is also proved by Circularity, but this time the circularity is released by applying the conditional **while**$_2$ axiom, and used in one of its conditions.

5.1 Soundness

The next result establishes the soundness of our proof system, in the sense of partial correctness. Note that, unlike the soundness of Hoare logic which is shown for each language separately, the soundness of reachability logic is proved only once, for all languages.

In order to prove soundness of the proof system, we need the following helper definition, which will allow us to make the proof by induction on the termination proof of g.

Definition 12. *Let $g \in \mathcal{T}_{Cfg}$ be an arbitrary configuration. We say that the unconditional reachability rule $\varphi \Rightarrow \varphi'$ is $(g, \geq)$-**strongly-valid** (resp. $(g, \geq)$-**strictly-strongly-valid**) if for all γ such that $g \geq \gamma$ and for all valuations ρ such that $(\gamma, \rho) \models \varphi$, there exists γ' such that $\gamma \to^{\star}_{\mathcal{S}} \gamma'$ (resp. $\gamma \to^{+}_{\mathcal{S}} \gamma'$) and $(\gamma', \rho) \models \varphi'$.*

We write $\mathcal{S} \models^{*}_{g\geq} \varphi \Rightarrow \varphi'$ when $\varphi \Rightarrow \varphi'$ is $(g, \geq)$-strongly-valid and $\mathcal{S} \models^{+}_{g\geq} \varphi \Rightarrow \varphi'$ if $\varphi \Rightarrow \varphi'$ is $(g, \geq)$-strictly-strongly-valid.

Intuitively, "$(g, \succeq)$-strongly-valid" is similar to "strongly valid", but only concerns configurations less than g, according to the termination dependence relation. If g terminates, then "$(g, \succeq)$-strongly-valid" is similar to "valid". The following lemma captures the link between the two notions:

Proposition 4. *$\mathcal{S} \models \varphi \Rightarrow \varphi'$ if and only if, for all terminating configurations $g \in \mathcal{T}_{Cfg}$, $\mathcal{S} \models^{*}_{g \succeq} \varphi \Rightarrow \varphi'$.*

Proof. We prove each implication separately.

"$\rightarrow$" Assume $\mathcal{S} \models \varphi \Rightarrow \varphi'$. We show that for all terminating $g \in \mathcal{T}_{Cfg}$, $\varphi \Rightarrow \varphi'$ is $(g, \succeq)$-strongly-valid. Let g be an arbitrary terminating configuration, let γ be an arbitrary configuration smaller or equal according to $\succeq$ than g (i.e. $g \succeq \gamma$) and let ρ be a valuation such that $(\gamma, \rho) \models \varphi$. As g terminates, it follows that γ also terminates. As $\mathcal{S} \models \varphi \Rightarrow \varphi'$, we have that there exists γ' such that $\gamma \rightarrow^{\star}_{\mathcal{S}} \gamma'$ and $(\gamma', \rho) \models \varphi'$. As γ was chosen arbitrarily such that $g \succeq \gamma$, it follows that $\varphi \Rightarrow \varphi'$ is $(g, \succeq)$-strongly-valid. As g was chosen arbitrarily such that it is terminating, it follows that for all terminating g, $\varphi \Rightarrow \varphi'$ is $(g, \succeq)$-strongly-valid, which is what we had to show.

"$\leftarrow$" Assume that for all terminating $g \in \mathcal{T}_{Cfg}$, $\varphi \Rightarrow \varphi'$ is $(g, \succeq)$-strongly-valid. We show that $\mathcal{S} \models \varphi \Rightarrow \varphi'$. Let γ be an arbitrary terminating configuration and let ρ be an arbitrary valuation such that $(\gamma, \rho) \models \varphi$. Let $g = \gamma$. We have that $g \succeq \gamma$ and that g is terminating. Therefore, by the assumption that for all terminating g, $\varphi \Rightarrow \varphi'$ is $(g, \succeq)$-strongly-valid, we obtain that there exists γ' such that $\gamma \rightarrow^{\star}_{\mathcal{S}} \gamma'$ and $(\gamma', \rho) \models \varphi'$. As the terminating configuration γ and the valuation ρ were chosen arbitrarily, it follows that $\mathcal{S} \models \varphi \Rightarrow \varphi'$. □

The following helper lemma is the core of the soundness proof. It shows that each proof in our proof system is $(g, \succeq)$-strongly-valid by induction on the proof tree and on g.

Lemma 1. *For any proof tree concluding $\mathcal{A} \vdash_C \varphi \Rightarrow \varphi'$, for all terminating configurations $g \in \mathcal{T}_{Cfg}$, if the conditional rules in $\mathcal{A}$ are weakly well-defined, if the unconditional rules in $\mathcal{A}$ are $(g, \succeq)$-strictly-strongly-valid and if C is $(g_0, \succeq)$-strictly-strongly-valid for all g_0 such that $g \succ g_0$, we have that:*

1. if C is empty, then $\varphi \Rightarrow \varphi'$ is $(g, \succeq)$-strongly-valid and
2. if C is not empty, then $\varphi \Rightarrow \varphi'$ is $(g, \succeq)$-strictly-strongly-valid.

Proof. By induction on the proof tree and case analysis on the last rule in the proof tree:

1. If the last rule is *Axiom*, let g be an arbitrary configuration and assume that the conditional rules in $\mathcal{A}$ are weakly well-defined, that the unconditial rules in $\mathcal{A}$ are $(g, \succeq)$-strictly-strongly-valid and that C is $(g_0, \succeq)$-strictly-strongly-valid for all configurations $g \succ g_0$. We show that $\varphi \wedge \psi \Rightarrow \varphi' \wedge \psi$ is $(g, \succeq)$-strictly-strongly-valid (this is the stronger conclusion of the two cases; $(g, \succeq)$-strictly-strongly-valid implies $(g, \succeq)$-strongly-valid for the case where C is empty). We distinguish two cases:

   (a) If $n > 0$, let γ be an arbitrary configuration such that $g \succeq \gamma$ and let ρ be an arbitrary valuation such that $(\gamma, \rho) \models \varphi \wedge \psi$. We show that there exists γ' such that $\gamma \rightarrow^{+}_{\mathcal{S}} \gamma'$ and $(\gamma', \rho) \models \varphi' \wedge \psi$. We first show that $\varphi_i \Rightarrow \varphi'_i$ is ρ-strongly-valid for all $1 \leq i \leq n$.

   Let $\gamma_1, \ldots, \gamma_n$ be arbitrary configurations such that $(\gamma_i, \rho) \models \varphi_i$ for all $1 \leq i \leq n$. As the rule is weakly well-defined, $\gamma_1, \ldots, \gamma_n$ exist. As ψ is stateless, it follows that $(\gamma_i, \rho) \models \varphi_i \wedge \psi$ for all $1 \leq i \leq n$.

   By induction on $1 \leq i \leq n$, we show that $\gamma \succ \gamma_i$ and that $\varphi_j \Rightarrow \varphi'_j$ is ρ-strongly-valid for all $1 \leq j < i$.

   Let $1 \leq i \leq n$ be fixed. By choice of γ_i, we have that $(\gamma_i, \rho) \models \varphi_i \wedge \psi$. By the induction hypothesis, we have that $\varphi_j \Rightarrow \varphi'_j$ is ρ-strongly-valid for all $1 \leq j < i$. Therefore, by the o.t. hypothesis, we have that $\gamma \succ \gamma_i$.

   As the unconditional rules in $\mathcal{A}$ are $(g, \succeq)$-strictly-strongly-valid and C is $(g_0, \succeq)$-strictly-strongly-valid for any configuration $g \succ g_0$, it follows that the unconditional rules in $\mathcal{A} \cup C$ are $(g_0, \succeq)$-strictly-strongly-valid for any configuration $\gamma \succ g_0$. In particular, the unconditional rules in $\mathcal{A} \cup C$ are $(\gamma_i, \succeq)$-strictly-strongly-valid. By the (outer) induction hypothesis, we obtain that $\varphi_i \wedge \psi \Rightarrow \varphi'_i$ is $(\gamma_i, \succeq)$-strongly-valid; therefore there exists γ'_i such that $\gamma_i \rightarrow^{\star}_{\mathcal{S}} \gamma'_i$ and $(\gamma'_i, \rho) \models \varphi'_i$. As γ_i was chosen arbitrarily, it follows that $\varphi_i \Rightarrow \varphi'_i$ is ρ-strongly-valid, which is what we had to prove.

   We have shown by induction on i that $\varphi_i \Rightarrow \varphi'_i$ is ρ-strongly-valid for all $1 \leq i \leq n$. Therefore, by the well-definedness of the conditional rule, we obtain that there exists γ' such that $\gamma \rightarrow^{+}_{\mathcal{S}} \gamma'$ and $(\gamma', \rho) \models \varphi'$. As $(\gamma, \rho) \models \varphi \wedge \psi$, it follows that $(\gamma, \rho) \models \psi$; as ψ is stateless, it follows that $(\gamma', \rho) \models \psi$. As $(\gamma', \rho) \models \varphi'$ and $(\gamma', \rho) \models \psi$, it follows that $(\gamma', \rho) \models \varphi' \wedge \psi$. We have shown that there exists γ' such that $\gamma \rightarrow^{+}_{\mathcal{S}} \gamma'$ and $(\gamma', \rho) \models \varphi' \wedge \psi$, which is what we had to show.

   (b) If $n = 0$, let γ be an arbitrary configuration such that $g \succeq \gamma$ and let ρ be an arbitrary valuation such that $(\gamma, \rho) \models \varphi \wedge \psi$. We show that there exists γ' such that $\gamma \rightarrow^{+}_{\mathcal{S}} \gamma'$ and $(\gamma', \rho) \models \varphi' \wedge \psi$.

   From $(\gamma, \rho) \models \varphi \wedge \psi$ we immediately obtain $(\gamma, \rho) \models \varphi$. As $\varphi \Rightarrow \varphi'$ is in $\mathcal{A}$, it must be, by hypothesis, $(g, \succeq)$-strictly-strongly-valid. Therefore there exists γ' such that $\gamma \rightarrow^{+}_{\mathcal{S}} \gamma'$ and $(\gamma', \rho) \models \varphi'$. As $(\gamma', \rho) \models \varphi' \wedge \psi$, we have $(\gamma, \rho) \models \psi$; as ψ is stateless, it follows that $(\gamma', \rho) \models \psi$. We already have that $(\gamma', \rho) \models \varphi'$ and therefore $(\gamma', \rho) \models \varphi' \wedge \psi$, which is what we had to show.

2. If the last rule is *Reflexivity*, let g be an arbitrary configuration. We show that $\varphi \Rightarrow \varphi$ is $(g, \succeq)$-strongly-valid. Let $g \succeq \gamma$ be an arbitrary configuration and let ρ be an arbitrary valuation such that $(\gamma, \rho) \models \varphi$. We show that there exists γ' such that $\gamma \rightarrow^{\star}_{\mathcal{S}} \gamma'$ and $(\gamma', \rho) \models \varphi$. Indeed, it is sufficient to choose $\gamma' = \gamma$ and the conclusion trivially follows. As C is empty, this is the only case to consider.

3. If the last rule is *Transitivity*, we distinguish two cases:

   (a) If C is empty, let g be an arbitrary configuration. We show that $\varphi_1 \Rightarrow \varphi_3$ is $(g, \succeq)$-strongly-valid. By the induction hypothesis we have that $\varphi_1 \Rightarrow \varphi_2$ and $\varphi_2 \Rightarrow \varphi_3$ are $(g, \succeq)$-strongly-valid.

   Let $g \succeq \gamma_1$ be an arbitrary configuration and let ρ be an arbitrary valuation such that $(\gamma_1, \rho) \models \varphi_1$. We show that there exists γ_3 such that $\gamma_1 \rightarrow^{\star}_{\mathcal{S}} \gamma_3$ and $(\gamma_3, \rho) \models \varphi_3$.

   As $\varphi_1 \Rightarrow \varphi_2$ is $(g, \succeq)$-strongly-valid, it follows that there exists γ_2 such that $\gamma_1 \rightarrow^{\star}_{\mathcal{S}} \gamma_2$ and $(\gamma_2, \rho) \models \varphi_2$. As $\varphi_2 \Rightarrow \varphi_3$ is $(g, \succeq)$-strongly-valid, it follows that there exists γ_3 such that $\gamma_2 \rightarrow^{\star}_{\mathcal{S}} \gamma_3$ and $(\gamma_3, \rho) \models \varphi_3$. In conclusion $\gamma_1 \rightarrow^{\star}_{\mathcal{S}} \gamma_3$ and $(\gamma_3, \rho) \models \varphi_3$, which is what we had to show.

(b) If C is not empty, let g be an arbitrary configuration. We show that $\varphi_1 \Rightarrow \varphi_3$ is $(g, \succeq)$-strictly-strongly-valid.

Let $g \succeq \gamma_1$ be an arbitrary configuration and let ρ be an arbitrary valuation such that $(\gamma_1, \rho) \models \varphi_1$. We show that there exists γ_3 such that $\gamma_1 \to_{\mathcal{S}}^{+} \gamma_3$ and $(\gamma_3, \rho) \models \varphi_3$.

By the induction hypothesis, we have that $\varphi_1 \Rightarrow \varphi_2$ is $(g, \succeq)$-strictly-strongly-valid. Therefore, there exists γ_2 such that $\gamma_1 \to_{\mathcal{S}}^{+} \gamma_2$ and $(\gamma_2, \rho) \models \varphi_2$.

Also by the induction hypothesis, as the unconditional rules in $\mathcal{A} \cup C$ are $(g_0, \succeq)$-strictly-strongly-valid for any $g \succ g_0$, it follows that $\varphi_2 \Rightarrow \varphi_3$ is $(g_0, \succeq)$-strongly-valid for any $g \succ g_0$. In particular $\varphi_2 \Rightarrow \varphi_3$ is $(\gamma_2, \succeq)$-strongly-valid. Therefore, there exists γ_3 such that $\gamma_2 \to_{\mathcal{S}}^{\star} \gamma_3$ and $(\gamma_3, \rho) \models \varphi_3$. In conclusion, $\gamma_1 \to_{\mathcal{S}}^{+} \gamma_2 \to_{\mathcal{S}}^{\star} \gamma_3$ and $(\gamma_3, \rho) \models \varphi_3$, which is what we had to show.

4. If the last rule is *Consequence*, let g be an arbitrary configuration. We show that $\varphi_1 \Rightarrow \varphi_2$ is $(g, \succeq)$-strongly-valid (resp. $(g, \succeq)$-strictly-strongly-valid).

   Let $g \succeq \gamma_1$ be an arbitrary configuration and let ρ be an arbitrary valuation such that $(\gamma_1, \rho) \models \varphi_1$. We show that there exists γ_2 such that $\gamma_1 \to_{\mathcal{S}}^{\star} \gamma_2$ (resp. $\gamma_1 \to_{\mathcal{S}}^{+} \gamma_2$) and $(\gamma_2, \rho) \models \varphi_2$.

   As $\models \varphi_1 \to \varphi_2$, we have that $(\gamma_1, \rho) \models \varphi_1'$. By the induction hypothesis, $\varphi_1' \Rightarrow \varphi_2'$ is $(g, \succeq)$-strongly-valid (resp. $(g, \succeq)$-strictly-strongly-valid) and therefore there exists γ_2 such that $\gamma_1 \to_{\mathcal{S}}^{\star} \gamma_2$ (resp. $\gamma_1 \to_{\mathcal{S}}^{+} \gamma_2$) and $(\gamma_2, \rho) \models \varphi_2'$. As $\models \varphi_2' \to \varphi_2$, it follows that $(\gamma_2, \rho) \models \varphi_2$, which is what we had to show.

5. If the last rule is *Case analysis*, let g be an arbitrary configuration. We show that $\varphi_1 \vee \varphi_2 \Rightarrow \varphi$ is $(g, \succeq)$-strongly-valid (resp. $(g, \succeq)$-strictly-strongly-valid).

   Let $g \succeq \gamma_1$ be an arbitrary configuration and let ρ be an arbitrary valuation such that $(\gamma_1, \rho) \models \varphi_1 \vee \varphi_2$. We show that there exists γ_2 such that $\gamma_1 \to_{\mathcal{S}}^{\star} \gamma_2$ (resp. $\gamma_1 \to_{\mathcal{S}}^{+} \gamma_2$) and $(\gamma_2, \rho) \models \varphi$.

   As $(\gamma_1, \rho) \models \varphi_1 \vee \varphi_2$ it follows that there exists $i \in \{1, 2\}$ such that $(\gamma_1, \rho) \models \varphi_i$. By the induction hypothesis, we have that $\varphi_i \Rightarrow \varphi$ is $(g, \succeq)$-strongly-valid (resp. $(g, \succeq)$-strictly-strongly-valid). Therefore, there exists γ_2 such that $\gamma_1 \to_{\mathcal{S}}^{\star} \gamma_2$ (resp. $\gamma_1 \to_{\mathcal{S}}^{+} \gamma_2$) and $(\gamma_2, \rho) \models \varphi$. But this is exactly what we had to show.

6. If the last rule is *Abstraction*, let g be an arbitrary configuration. We show that $\exists X. \varphi \Rightarrow \varphi'$ is $(g, \succeq)$-strongly-valid (resp. $(g, \succeq)$-strictly-strongly-valid).

   Let $g \succeq \gamma$ be an arbitrary configuration and let ρ be an arbitrary configuration such that $(\gamma, \rho) \models \exists X. \varphi$. We show that there exists γ' such that $\gamma \to_{\mathcal{S}}^{\star} \gamma'$ (resp. $\gamma \to_{\mathcal{S}}^{+} \gamma'$) and $(\gamma', \rho) \models \varphi'$.

   As $(\gamma, \rho) \models \exists X. \varphi$, it follows that there exists a valuation ρ' which differs from ρ only in X such that $(\gamma, \rho') \models \varphi$. By the induction hypothesis ($\varphi \Rightarrow \varphi'$ is $(g, \succeq)$-strongly-valid (resp. $(g, \succeq)$-strictly-strongly-valid)), it follows that there exists γ' such that $\gamma \to_{\mathcal{S}}^{\star} \gamma'$ (resp. $\gamma \to_{\mathcal{S}}^{+} \gamma'$) and $(\gamma', \rho') \models \varphi'$. As X contains no free variable of φ' and ρ differs from ρ' only in X, it follows that $(\gamma', \rho) \models \varphi'$, which is what we had to show.

7. If the last rule is *Circularity*, we show by (an inner) induction on g that:

   > If the unconditional rules in $\mathcal{A}$ are $(g, \succeq)$-strictly-strongly-valid and if C is strictly $(g_0, \succeq)$-strictly-strongly-validfor all $g \succ g_0$, then $\varphi \Rightarrow \varphi'$ is $(g, \succeq)$-strictly-strongly-valid.

   Assume that the unconditional rules in $\mathcal{A}$ are $(g, \succeq)$-strongly-valid and that C is $(g_0, \succeq)$-strictly-strongly-validfor any $g \succ g_0$.

   By the (inner) induction hypothesis, we know that $\varphi \Rightarrow \varphi'$ is $(g_0, \succeq)$-strictly-strongly-valid for any $g \succ g_0$. Therefore $C \cup \{\varphi \Rightarrow \varphi'\}$ is $(g_0, \succeq)$-strictly-strongly-validfor any $g \succ g_0$. Therefore, by the (outer) induction hypothesis, we obtain that $\varphi \Rightarrow \varphi'$ is $(g, \succeq)$-strictly-strongly-valid, which is what we had to show. □

Using the helper lemma that we have proved above, we are ready to show that the proof system is sound:

Theorem 1 (Soundness). *If $\mathcal{S}$ is a weakly well-defined reachability system, then $\mathcal{S} \vdash \varphi \Rightarrow \varphi'$ implies $\mathcal{S} \models \varphi \Rightarrow \varphi'$.*

Proof. By Proposition 4 and Lemma 1. □

Because of the utmost importance of the result above, we have also mechanized its proof. Our complete Coq formalization can be found at `http://fsl.cs.uiuc.edu/RL`. The proof is parametric in the operational semantics $\mathcal{S}$ and thus can be used to produce formal correctness certificates for program verification tasks. The URL above also includes a human readable proof of this result, as well as several derived proof rules which are useful for verifying programs, together with their soundness proofs, such as weakening, logic framing, set circularity, and substitution. Set circularity allows introducing several circularities in advance, rather than just one when that situation occurs as a subgoal of a proof. This rule is at the core of the reachability logic MatchC prover [34], where is useful for proving properties about mutually recursive functions.

Our reachability proof system can also prove divergence:

Corollary 1. *If $\mathcal{S}$ is also ω-closed, then $\mathcal{S} \vdash \varphi \Rightarrow \omega$ implies $\mathcal{S} \models \varphi\uparrow$.*

5.2 Derived Rules

Now we present several useful proof rules which are consequences of the system above.

The first derived rule is weakening, which shows that a derivation can still be carried out with "stronger" assumptions. With separate sets of axioms and circularities, stronger should include promoting a hypothesis from the circularities to the axioms. Adding additional circularities may invalidate a use of the reflexivity rule, so this can be permitted only if the circularities are non-empty. We also include logical implication betwen rules, also taking into account the framing provided by the Axiom rule. This is usually not required but sometimes indespensible.

For this generalization of inclusion, we use the notation $\sqsubseteq$. $\mathcal{X} \sqsubseteq \mathcal{X}'$ when any conditional rules of $\mathcal{X}$ also occur in $\mathcal{X}'$ and for every unconditional rule $\varphi \Rightarrow \varphi'$ in $\mathcal{X}$ there is some rule $\varphi_1 \Rightarrow \varphi_1'$ in $\mathcal{X}'$ and some structureless ψ such that $\models \varphi \to \varphi_1 \wedge \psi$ and $\models \varphi_1' \wedge \psi \to \varphi'$.

Lemma 2 (Weakening). *If $\mathcal{A} \vdash_C \varphi \Rightarrow \varphi'$, then $\mathcal{A}' \vdash_{C'} \varphi \Rightarrow \varphi'$ for any C' and $\mathcal{A}'$ with $\mathcal{A} \sqsubseteq \mathcal{A}'$, $C \sqsubseteq C' \cup \mathcal{A}'$, and $C' = \emptyset$ if $C = \emptyset$.*

Proof. By structural induction. In the axiom case, if a conditional rule was used then $\mathcal{A}'$ must have the same rule, and if an unconditional rule was used then $\mathcal{A}'$ at least has a rule that implies the conclusion when framed by some additional ψ_0. If the original application of the axiom rule used a framing condition ψ, the new proof can frame by $\psi \wedge \psi_0$. If the original proof used reflexivity, then C is empty and so by assumption C' is empty as well, and the new proof can use the reflexivity rule. In the circularity rule, note that adding a new assumption to C gives a nonempty set and preserves $\sqsubseteq$. In the transitivity rule the second premise is a proof with $\mathcal{A} \cup C$ as axioms and empty circularities. By simple reasoning about $\sqsubseteq$, $\mathcal{A} \cup C \sqsubseteq \mathcal{A}' \cup C'$, so the new proof can proceed by applying the transitivity rule to the inductive hypotheses. The remaining cases are similarly straightforward. □

Both weakening lemmas of [34] are instances of this rule. It also allows us to prove the following important result, which justifies local reasoning in structureless contexts. We call it logical framing, to emphasize that the frame only contains logical constraints over the rule's variables. Logical framing was a proof rule in itself in [32–34]. To make logical facts available to the proofs for the premises of a conditional rule it was necessary to allow framing in the Axiom rule, from which we can derive the general rule.

Lemma 3 (**Logical Framing**). *If $\mathcal{A} \vdash_C \varphi \Rightarrow \varphi'$ and ψ is any structureless formula, $\mathcal{A} \vdash_C \varphi \land \psi \Rightarrow \varphi' \land \psi$.*

Proof. The proof proceeds in two steps, first showing that $\varphi \land \psi \Rightarrow \varphi' \land \psi$ is true under the modified context which also frames ψ onto all the unconditional rules in the $\mathcal{A}$ and C, and then applying weakening to conclude $\varphi \land \psi \Rightarrow \varphi' \land \psi$ is also provable under the original context.

For the first step, use the notation $\mathcal{X} \land \psi$ for the set of assumptions obtained by framing ψ onto the unconditional rules of $\mathcal{X}$.

$$\mathcal{X} \land \psi = \{\varphi \land \psi \Rightarrow \varphi' \land \psi \mid \varphi \Rightarrow \varphi' \in \mathcal{X}\}$$
$$\cup \{\mu \mid \mu \in \mathcal{X}, \mu \text{ is a conditional rule}\}$$

With this notation, the goal is to prove $\mathcal{A} \land \psi \vdash_{C \land \psi} \varphi \land \psi \Rightarrow \varphi' \land \psi$. The proof proceeds by induction on the depth of the proof tree $\mathcal{A} \vdash_C \varphi \Rightarrow \varphi'$. Consider the last rule used in the proof.

1. If the rule is an axiom rule, then φ and φ' have the form $\varphi_0 \land \psi_0$ and $\varphi'_0 \land \psi_0$ for some ψ_0 chosen in the application of the rule. If a conditional axiom "$\varphi \Rightarrow \varphi'$ if $\varphi_i \Rightarrow \varphi'_i$ was used, $\mathcal{A} \land \psi$ contains the same conditional axiom. Begin the new proof by using consequence to reassociate the conjuctions in the goal, and apply the axiom rule chosing the same axiom and using $\psi_0 \land \psi$ as the framing condition. The inductive hypotheses are $(\mathcal{A} \cup C) \land \psi \vdash_{\emptyset \land \psi} (\varphi_i \land \psi_0) \land \psi \Rightarrow \varphi'_i \land \psi$ for each i, and the requried premises of the axiom rule are $(\mathcal{A} \land \psi) \cup (C \land \psi) \vdash \varphi_i \land (\land \psi_0 \psi) \Rightarrow \varphi'_i$. By calculation the sets of assumptions are identical, and the formulas can be aligned by applying the consequence rule to reassociate conjunctions in the left formula and drop the extra condition in the right formula. If the axiom rule applied an unconditional rule $\varphi_0 \Rightarrow \varphi'_0$ with framing condition ψ_0, then the set of assumptions $\mathcal{A} \land \psi$ will contain the rule $\varphi_0 \land \psi \Rightarrow \varphi'_0 \land \psi$. This can be applied with framing condition ψ_0 to prove $(\varphi_0 \land \psi) \land \psi_0 \Rightarrow (\varphi'_0 \land \psi) \land \psi$.
2. If the last rule was reflexivity, the proof concludes also with reflexivity, justified by the observation above that $\land \psi$ is absorbed by $\emptyset$.
3. If the last rule was an application of transitivity the proof proceeds by applying transitivity to the inductive hypothesis, using both the observations above that $\land \psi$ is absorbed by $\emptyset$ and distributes over $\cup$ to justify using the inductive hypothesis for the second premise of the transitivity rule.
4. If the last rule was consequence, the proof beings with a single application of consequence, noting that $\varphi \land \psi \to \varphi' \land \psi$ is a valid formula whenever $\varphi \to \varphi'$ is.
5. If the last rule was case, the new proof beings by using consequence to rearrange the goal from $(\varphi_1 \land \varphi_2) \land \psi \Rightarrow \varphi' \land \psi$ into $(\varphi_1 \land \psi) \land (\varphi_2 \land \psi) \Rightarrow \varphi' \land \psi$, and can then apply the case rule using the induction hypotheses.
6. If the last rule was abstraction, then the conclusion has the form $(\exists x.\varphi) \Rightarrow \varphi'$. By renaming, which does not change the depth of a proof tree, we can ensure x in not free in ψ, obtain inductive hypothesis $\varphi \land \psi \Rightarrow \varphi' \land \psi$, and and finish by abstraction.
   The renaming lemma we need here applies a permutation π on variables to the conclusion of rules. A renaming operation ensures $(\gamma, \rho) \models \pi(\varphi) \Rightarrow \pi(\varphi')$ iff $(\gamma, \rho \circ \pi) \models \varphi \Rightarrow \varphi'$, and also that $\pi(\exists x.\varphi) = \exists \pi(x).\pi(varphi)$. The second condition may appear to require a very syntactic formula representation storing identifiers for bound variables, but note that it holds immediately for locally nameless syntax. The proof proceeds by induction. The second clause of the definition handles the abstraction case, the axiom case depends on the stipluation in section 5 that the axiom rule can apply an α renaming of any rule μ in $\mathcal{A}$. Other cases are straightforward, applying the original proof rule to the inductive hypothesis.
7. If the last rule of the original proof was circularity, the new proof can also apply circularitiy, and immediately use the inductive hypothesis.

□

The next lemma is a simple consequence of the abstraction rule.

Lemma 4. *If $\mathcal{A} \vdash_C \varphi \Rightarrow \varphi'$ then $\mathcal{A} \vdash_C \exists X.\varphi \Rightarrow \exists X.\varphi'$.*

Proof. Apply abstraction (X is not free in $\exists X.\varphi'$), and then consequence ($\models \varphi' \to \exists X.\varphi'$). □

The next lemma is Set Circularity , which allows introducing circularities in advance rather than just once a situation occurs as a subgoal of a proof, and also allows introducing several assumptions together. This rule is at the core of the reachability logic MatchC prover [34], where is useful for proving properties about mutually recursive functions:

Lemma 5 (**Set Circularity**). *If $\mathcal{A} \vdash_C \mu$ for each rule μ in a finite set of unconditional rules C, then also $\mathcal{A} \vdash \mu$ for each μ in C.*

Proof. To prove this lemma we will fix one particular rule to prove, and use a form of cut rule to eliminate one assumption at a time from C.

Suppose without loss of generality that none of the rules in C are already included in $\mathcal{A}$, if some rule was then the other proofs will still go through even without that rule in C.

Fix a particular proof $\mathcal{A} \vdash_C \varphi \Rightarrow \varphi'$, and proceed by induction on $|C|$. If C is empty the proof is finished. Otherwise, pick some $\varphi_1 \Rightarrow \varphi_2$ in C. By applying the cut lemma with each other proof, we can conclude that each rule in $C \setminus \{\varphi_1 \Rightarrow \varphi_2\}$ and the distinguished rule rule $\varphi \Rightarrow \varphi'$ are provable under $\mathcal{A} \vdash_{C \setminus \{\varphi_1 \Rightarrow \varphi_2\}}$, which has reduced the size of C by one.

The cut lemma concludes $\mathcal{A} \setminus \{\varphi_1 \Rightarrow \varphi_2\} \vdash_{C \setminus \{\varphi_1 \Rightarrow \varphi_2\}} \varphi \Rightarrow \varphi'$ given a proof $\mathcal{A} \vdash_C \varphi \Rightarrow \varphi'$ and a proof $\mathcal{A} \setminus \{\varphi_1 \Rightarrow \varphi_2\} \vdash_{C \cup \{\varphi_1 \Rightarrow \varphi_2\}} \varphi_1 \Rightarrow \varphi_2$.

The proof goes by structural induction on the proof of $\varphi \Rightarrow \varphi'$. Note the given subproof for $\varphi_1 \Rightarrow \varphi_2$ can be used under $C \setminus \{\varphi_1 \Rightarrow \varphi_2\}$ by first applying circularity, but has a manifestly non-empty set of circular assumptions which makes it easier to apply weakening over the course of the induction.

If the orginal proof used the axiom rule to prove $\varphi_1 \Rightarrow \varphi_2$ use the subproof instead. If the original proof used circularity to show $\varphi_1 \Rightarrow \varphi_2$, also use the given subproof. If the original proof concluded with reflexivity, note that removing $\varphi_1 \Rightarrow \varphi_2$ from the empty set leaves the empty set, so reflexivity is still valid. Every other case neither adds, removes, nor depends on finding $\{\varphi_1 \Rightarrow \varphi_2\}$ in any set of assumptions so they can proceed by applying the same proof rule, and weaken the subproof of $\{\varphi_1 \Rightarrow \varphi_2\}$ to match the assumptions of the inductive hypotheses. □

The next derived rule is substitution, which we only show for first-order matching logic. Stating this lemma for abstract matching logic requires an axiomatization of substitution for an abstract matching logic, which is included in our Coq proofs but not discussed here.

Lemma 6 (Substitution). *If $\mathcal{A} \vdash_C \varphi \Rightarrow \varphi'$ and θ is a substitution, then $\mathcal{A} \vdash_C \theta(\varphi) \Rightarrow \theta(\varphi')$.*

Proof. Consider the set of free variables $X = FV(\varphi) \cup FV(\varphi') \cup FV(\theta(\varphi)) \cup FV(\theta(\varphi'))$. Let x_i for $1 \le i \le n$ be an enumeration of this set. There are an infinite number of variables, so it is possible to find a set Y of n variables y_i which are not in X. Consider the formulas $\theta(\varphi)$ and $\varphi_\theta = \exists Y.(\bigwedge_{1\le i\le n} y_i = \theta(x_i)) \wedge \exists X.(\bigwedge_{1\le i\le n} x_i = y_i) \wedge \varphi$.

By well-definedness, for any ρ there are unique γ_i such that $(\rho, \gamma_i) \models \theta(x_i)$. By properties of pattern matching, $(\gamma, \rho) \models \varphi_\theta$ iff $(\gamma, \rho[y_i \mapsto \gamma_i]) \models \exists X.(\bigwedge_{1\le i\le n} x_i = y_i) \wedge \varphi$ iff $(\gamma, \rho[y_i \mapsto \gamma_i, x_i \mapsto \gamma_i]) \models \varphi$. By choice of Y, the valuation here agrees with $\rho \circ \theta$ on X, so by the properties of free variables this holds iff $(\gamma, \rho \circ \theta) \models \varphi$. By the properties of substition, this holds iff $(\gamma, \rho) \models \theta(\varphi)$, so $\theta(\varphi)$ and φ_θ are logically equivalent. By the same argument, $\theta(\varphi')$ is logically equivalent to a similarly constructed φ'_θ.

With this observation, $\mathcal{A} \vdash_C \theta(\varphi) \Rightarrow \theta(\varphi')$ can be proven by applying the consequence rule to leave the goal $\mathcal{A} \vdash_C \varphi_\theta \Rightarrow \varphi'_\theta$, then repeatedly applying abstraction and framing to remove the quantification and the structureless formulas $\bigwedge_{1\le i\le n} y_i = \theta(x_i)$ and $\bigwedge_{1\le i\le n} x_i = y_i$. This leaves the goal $\mathcal{A} \vdash_C \varphi \Rightarrow \varphi'$ for which we have a proof. $\square$

6. Relative Completeness

Here we show that the reachability logic proof system is relatively complete. This means that any valid reachability property of any programming language (or calculus, system, etc.) is formally derivable with our fixed language-independent proof system in Figure 3 using the operational semantics rules of the language as axioms. Note that this is a stronger result than the relative completeness of Hoare logics, since the latter needs to be proved for each language separately, taking into account its particularities. We prove our relative completeness result once and for all languages, similarly to our soundness proof. Relativity here refers to the configuration model: since a matching logic includes a configuration model and since that model can comprise arbitrarily complex domains, we assume an oracle capable of answering FOL validity questions on that model.

While in Section 5 we proved soundness for any abstract matching logic, for completeness we need a precise constructive description of patterns and reachability rules, that is, a concrete matching logic instance. The obvious choice is first-order matching logic. Additionally, we assume the following

Framework:
The configuration signature Σ has
— a sort $\mathbb{N}$ with constants 0, 1 and binary operations $+$ and $\times$;
— an operation symbol $\alpha : Cfg \to \mathbb{N}$.
The configuration model $\mathcal{T}$ interprets
— $\mathbb{N}$ as the natural numbers and $0, 1, +, \times$ correspondingly;
— α as an injective function from $\mathcal{T}_{Cfg}$ to $\mathbb{N}$.
The reachability system $\mathcal{S}$ is
— non-empty, finite, well-defined, and ω-closed;
— its termination dependence relation $\succ$ is finitely branching: for each $\gamma \in \mathcal{T}_{Cfg}$ there are finitely many $\gamma' \in \mathcal{T}_{Cfg}$ with $\gamma \succ \gamma'$.

The assumptions on $\mathbb{N}$ and its operations are needed in order to express Gödel's β predicate. Those on α are needed in order to enumerate the configurations. Gödelization and "state" enumeration are expected assumptions for any relative completeness result. Moreover, operational semantics are expected to consist of non-empty finite sets of well-defined rules. $\mathcal{S}$ is assumed ω-closed in order to derive the divergence of configurations with computations with infinite nesting of conditions (like in the case of a big-step semantics). As discussed in Section 4.3, it is easy to construct the ω-closure $\mathcal{S}^\omega$, which yields the same transition system as $\mathcal{S}$. The finite branching of the termination dependence relation generalizes finite branching of the transition relation $\to_\mathcal{S}$ (a completeness requirement in [34]), for the case where $\mathcal{S}$ only contains unconditional rules.

Our proof proceeds as follows: we first show that first-order matching logic pattern reasoning reduces to FOL reasoning in the model $\mathcal{T}$; then we encode transition system operations in FOL, making use of Gödel's β predicate to eliminate quantifications over sequences of configurations (needed for expressing reachability); then we show that semantic validity of reachability rules can also be expressed in FOL; finally, we prove our completeness result.

6.1 Embedding First-Order Matching Logic in FOL

Here we recall how pattern reasoning in first-order matching logic reduces to FOL reasoning in the configuration model $\mathcal{T}$ [32–35]:

Definition 13. *Let $\square$ be a fresh Cfg variable (i.e., not in Var), and let $Var^\square$ be the extended set of variables $Var \cup \{\square\}$. For a pattern φ, let $\varphi^\square$ be the FOL formula obtained by replacing basic patterns $\pi \in T_{\Sigma,Cfg}(Var)$ with equalities $\square = \pi$. If $\gamma \in \mathcal{T}_{Cfg}$ and $\rho : Var \to \mathcal{T}$, then let $\rho^\gamma : Var^\square \to \mathcal{T}$ extend ρ by mapping $\square$ into γ: $\rho^\gamma(\square) = \gamma$ and $\rho^\gamma(x) = \rho(x)$ for all $x \in Var$. To highlight the semantic indistinguishability between patterns with variables in Var and the corresponding fragment of FOL with variables in $Var^\square$, we take the freedom to write $(\gamma, \rho) \models \varphi^\square$ in the FOL fragment, too, instead of $\rho^\gamma \models \varphi^\square$. A matching logic (respectively FOL) formula ψ is **structureless** (called "patternless" in [32, 33]) iff it contains no basic pattern (respectively no $\square$ variable), that is, iff $\psi = \psi^\square$.*

The notation in Definition 13 is consistent: if φ is a pattern, $\gamma \in \mathcal{T}_{Cfg}$ and $\rho : Var \to \mathcal{T}$, then $(\gamma, \rho) \models \varphi$ iff $(\gamma, \rho) \models \varphi^\square$. Also $\models \varphi$ iff $\mathcal{T} \models \varphi^\square$. Therefore, patterns form a methodological fragment of the FOL theory of $\mathcal{T}$, so we can use conventional theorem provers or proof assistants for pattern reasoning.

It is often technically convenient to eliminate $\square$ from a formula $\varphi^\square$ corresponding to a pattern φ. This can be done by replacing $\square$ with a configuration variable $c \in Var_{Cfg}$ (perhaps avoiding the free variables of φ). Indeed, $\varphi^\square[c/\square]$ is structureless. Then we have the following result:

Lemma 7. $(\rho(c), \rho) \models \varphi^\square$ *iff* $\rho \models \varphi^\square[c/\square]$.

Proof. With the notation in Definition 13, $(\rho(c), \rho) \models \varphi^\square$ iff $\rho^{\rho(c)} \models \varphi^\square$. Notice that if a valuation agrees on two variables, then it satisfies a formula iff it satisfies the formula obtained by substituting one of the two variables for the other. In particular, since $\rho^{\rho(c)}(\square) = \rho^{\rho(c)}(c)$, it follows that $\rho^{\rho(c)} \models \varphi^\square$ iff $\rho^{\rho(c)} \models \varphi^\square[c/\square]$. We notice that $\square$ does not occur in $\varphi^\square[c/\square]$, thus $\rho^{\rho(c)} \models \varphi^\square[c/\square]$ iff $\rho \models \varphi^\square[c/\square]$, and we are done. $\square$

Well-defined patterns have the following property, which we use extensively in Section 6.4:

Lemma 8. *If φ is well-defined, then $\models \varphi^\square \wedge \varphi^\square[c/\square] \to \square = c$.*

Proof. Let $\gamma \in \mathcal{T}_{Cfg}$ and $\rho : Var \to \mathcal{T}$. It suffices to prove that if $(\gamma, \rho) \models \varphi^\square$ and $(\gamma, \rho) \models \varphi^\square[c/\square]$ then $(\gamma, \rho) \models \square = c$. Since $\varphi^\square[c/\square]$ is structureless, we have that $(\gamma, \rho) \models \varphi^\square[c/\square]$ iff $\rho \models \varphi^\square[c/\square]$. By Lemma 7 that is iff $(\rho(c), \rho) \models \varphi^\square$. Further, since φ is well-defined, by Definition 3 there exists precisely one γ such that $(\gamma, \rho) \models \varphi^\square$, thus $\gamma = \rho(c)$. Then we can conclude that $(\gamma, \rho) \models \square = c$, and we are done. $\square$

6.2 Encoding Transition System Operations in FOL

Recall that one of the assumption of relative completeness is that $\mathcal{S}$ is well-defined (see Definition 9). For the purposes of this proof, we give an equivalent and finer-grained definition of the transition relation $\to_\mathcal{S}$ induced by a well-defined reachability system $\mathcal{S}$. Let

$k, m \in \mathbb{N}$. Recall from Definition 5 that $\mathcal{R}_k$ is the transition relation obtained by applying at most $k-1$ "nested" conditional rules. We introduce $\mathcal{R}_{k,m}$ with $\mathcal{R}_{k,m} \subseteq \mathcal{R}_k \subseteq \mathcal{T}_{Cfg} \times \mathcal{T}_{Cfg}$ to denote the transition relation obtained by applying at most $k-1$ "nested" conditional rules, and by taking at most m steps in each condition. Formally,

- $\mathcal{R}_{0,m} = \emptyset$
- $\mathcal{R}_{k+1,m} = \{ (\gamma,\ \gamma') \mid$ there exists some reachability rule

$$\varphi \Rightarrow \varphi' \text{ if } \varphi_1 \Rightarrow \varphi'_1 \wedge \ldots \wedge \varphi_n \Rightarrow \varphi'_n$$

in $\mathcal{S}$ and some valuation $\rho\colon Var \to \mathcal{T}$ such that:

1. $(\gamma,\rho) \models \varphi$ and $(\gamma',\rho) \models \varphi'$; and
2. there exist $\gamma_1, \ldots \gamma_n, \gamma'_1, \ldots, \gamma'_n \in \mathcal{T}_{Cfg}$ with $(\gamma_i\ ,\rho) \models \varphi_i$ and $(\gamma'_i\ ,\rho) \models \varphi'_i$ and such that $(\gamma_i,\ \gamma'_i) \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k,m}$ for all $1 \le i \le n$, where $\mathcal{R}^{m'}_{k,m}$ is the transitive composition of $\mathcal{R}_{k,m}$ with itself m' times ($\mathcal{R}^0_{k,m}$ is the identity)$\}$

Notice that $\mathcal{R}_{k,m}$ existentially quantifies $\gamma_1, \ldots, \gamma_n$ ("there exist $\gamma'_1, \ldots, \gamma'_n$"), unlike $\mathcal{R}_k$, which universally quantifies $\gamma_1, \ldots, \gamma_n$ ("for all $\gamma_1, \ldots, \gamma_n$"). However, the well-definedness of $\mathcal{S}$ implies that for a given ρ the said $\gamma_1, \ldots, \gamma_n$ always exist and are unique. Thus, we can replace universal with existential quantification. The following formally states that the relations $\mathcal{R}_{k,m}$ give an equivalent definition to the relations $\mathcal{R}_k$ and $\to_{\mathcal{S}}$:

Lemma 9. *$\mathcal{R}_k = \bigcup_{m\ge0} \mathcal{R}_{k,m}$ and $\to_{\mathcal{S}} = \bigcup_{k\ge0,m\ge0} \mathcal{R}_{k,m}$.*

Proof. First, we prove $\mathcal{R}_k = \bigcup_{m\ge0} \mathcal{R}_{k,m}$ by induction on k. The base case ($k = 0$) is trivial, as $\mathcal{R}_k = \emptyset$ and $\mathcal{R}_{k,m} = \emptyset$ for each $m \ge 0$. For the induction case, we assume the result for k and prove it for $k+1$ by double inclusion.

To show $\mathcal{R}_{k+1} \subseteq \bigcup_{m\ge0} \mathcal{R}_{k+1,m}$, we assume $(\gamma,\ \gamma') \in \mathcal{R}_{k+1}$ and we show $(\gamma,\ \gamma') \in \mathcal{R}_{k+1,m}$ for some m. By Definition 5, it follows that there exists some rule

$$\varphi \Rightarrow \varphi' \text{ if } \varphi_1 \Rightarrow \varphi'_1 \wedge \ldots \wedge \varphi_n \Rightarrow \varphi'_n$$

in $\mathcal{S}$ and some ρ such that

1. $(\gamma,\rho) \models \varphi$ and $(\gamma',\rho) \models \varphi'$; and
2. for all $\gamma_1, \ldots \gamma_n$ with $(\gamma_i\ ,\rho) \models \varphi_i$ for each $1 \le i \le n$ there exist $\gamma'_1, \ldots, \gamma'_n$ with $(\gamma'_i\ ,\rho) \models \varphi'_i$ such that $(\gamma_i,\ \gamma'_i) \in \mathcal{R}^*_k$ for each $1 \le i \le n$.

Since $\mathcal{S}$ is well-defined, it follows that there exist and are unique $\gamma_1, \ldots \gamma_n$ with $(\gamma_i\ ,\rho) \models \varphi_i$ for each $1 \le i \le n$, thus condition 2. above becomes: there exist $\gamma_1, \ldots \gamma_n,\ \gamma'_1, \ldots, \gamma'_n$ with $(\gamma_i\ ,\rho) \models \varphi_i$ and $(\gamma'_i\ ,\rho) \models \varphi'_i$ and such that $(\gamma_i,\ \gamma'_i) \in \mathcal{R}^*_k$ for all $1 \le i \le n$. Hence, there exist m_i and $\gamma_{i,0}, \ldots, \gamma_{i,m_i}$ with $\gamma_{i,0} = \gamma_i$ and $\gamma_{i,m_i} = \gamma'_i$ such that $(\gamma_{i,j}, \gamma_{i,j+1}) \in \mathcal{R}_k$ for each $1 \le i \le n$ and $0 \le j < m_i$. By the induction hypothesis, $\mathcal{R}_k = \bigcup_{m\ge0} \mathcal{R}_{k,m}$. Thus, there exist some $m_{i,0}, \ldots, m_{i,m_i-1}$ such that $(\gamma_{i,j}, \gamma_{i,j+1}) \in \mathcal{R}_{k,m_{i,j}}$ for each $1 \le i \le n$ and $0 \le j < m_i$. It is easy to prove by induction on k that $\mathcal{R}_{k,m'} \subseteq \mathcal{R}_{k,m''}$ if $m' \le m''$. Let m be the maximum of $\{m_i \mid 1 \le i \le n\} \cup \{m_{i,j} \mid 1 \le i \le n, 0 \le j \le m_i\}$. Then, it follows that $(\gamma_{i,j}, \gamma_{i,j+1}) \in \mathcal{R}_{k,m}$ for each $1 \le i \le n$ and $0 \le j < m_i$, and consequently, that $(\gamma_i, \gamma'_i) \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k,m}$. We can conclude that both conditions 1. and 2. in the definition of $\mathcal{R}_{k+1,m}$ are satisfied, that is, $(\gamma,\ \gamma') \in \mathcal{R}_{k+1,m}$ for some m. Therefore, $\mathcal{R}_{k+1} \subseteq \bigcup_{m\ge0} \mathcal{R}_{k+1,m}$.

To show $\bigcup_{m\ge0} \mathcal{R}_{k+1,m} \subseteq \mathcal{R}_{k+1}$, we assume $(\gamma,\ \gamma') \in \mathcal{R}_{k+1,m}$ for some m and we show $(\gamma,\ \gamma') \in \mathcal{R}_{k+1}$. By the definition of $\mathcal{R}_{k+1,m}$, it follows that there exists some rule

$$\varphi \Rightarrow \varphi' \text{ if } \varphi_1 \Rightarrow \varphi'_1 \wedge \ldots \wedge \varphi_n \Rightarrow \varphi'_n$$

in $\mathcal{S}$ and some ρ such that:

1. $(\gamma,\rho) \models \varphi$ and $(\gamma',\rho) \models \varphi'$; and
2. there exist $\gamma_1, \ldots \gamma_n, \gamma'_1, \ldots, \gamma'_n$ with $(\gamma_i, \rho) \models \varphi_i$ and $(\gamma'_i, \rho) \models \varphi'_i$ and such that $(\gamma_i,\ \gamma'_i) \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k,m}$ for all $1 \le i \le n$.

Since $\mathcal{S}$ is well-defined, it follows that there are unique $\gamma_1, \ldots \gamma_n$ with $(\gamma_i\ ,\rho) \models \varphi_i$ for each $1 \le i \le n$, thus condition 2. above becomes: for all $\gamma_1, \ldots \gamma_n$ with $(\gamma_i\ ,\rho) \models \varphi_i$ for each $1 \le i \le n$ there exist $\gamma'_1, \ldots, \gamma'_n$ with $(\gamma'_i\ ,\rho) \models \varphi'_i$ such that $(\gamma_i,\ \gamma'_i) \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k,m}$ for each $1 \le i \le n$. By the induction hypothesis, $\mathcal{R}_k = \bigcup_{m\ge0} \mathcal{R}_{k,m}$. Thus, $\mathcal{R}_{k,m} \subseteq \mathcal{R}_k$, and consequently, $\bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k,m} \subseteq \mathcal{R}^*_k$. We can conclude that both conditions 1. and 2. in Definition 5 are satisfied, that is, $(\gamma,\ \gamma') \in \mathcal{R}_{k+1}$. Therefore, $\bigcup_{m\ge0} \mathcal{R}_{k+1,m} \subseteq \mathcal{R}_{k+1}$.

The second part of the lemma follows from the first part, as $\to_{\mathcal{S}}$ is defined to be $\bigcup_{k\ge0} \mathcal{R}_k$. □

Since our objective is to encode properties of the transition system $(\mathcal{T}_{Cfg}, \to_{\mathcal{S}})$ in FOL making use of the relations $\mathcal{R}_{k,m}$, we next discuss these relations in a bit more detail. Unless otherwise specified, we fix some arbitrary $k, m \in \mathbb{N}$ and assume that each rule in $\mathcal{S}$ has at most nc conditions. Then $(\gamma, \gamma') \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k,m}$ iff there exist some $\gamma_0, \ldots, \gamma_{m'} \in \mathcal{T}_{Cfg}$ with $0 \le m' \le m$, $\gamma_0 = \gamma$ and $\gamma_{m'} = \gamma'$ such that $(\gamma_{i_1}, \gamma_{i_1+1}) \in \mathcal{R}_{k,m}$ for each $0 \le i_1 < m'$. We can reformulate it as there exist some $\gamma_0, \ldots, \gamma_m \in \mathcal{T}_{Cfg}$ with $\gamma_0 = \gamma$ and $\gamma_m = \gamma'$ such that $(\gamma_{i_1}, \gamma_{i_1+1}) \in \mathcal{R}_{k,m}$ or $\gamma_{i_1} = \gamma_{i_1+1}$ for each $0 \le i_1 < m$. A pair $(\gamma_{i_1}, \gamma_{i_1+1})$ belongs to $\mathcal{R}_{k,m}$ iff there exist some rule $\mu \equiv (\varphi \Rightarrow \varphi' \text{ if } \varphi_1 \Rightarrow \varphi'_1 \wedge \ldots \wedge \varphi_n \Rightarrow \varphi'_n)$ in $\mathcal{S}$ and valuation ρ_{i_1} such that for each $1 \le i_2 \le n$ there exist some $\gamma_{i_1,i_2,0}$ and $\gamma_{i_1,i_2,m}$ with $(\gamma_{i_1,i_2,0}, \rho_{i_1}) \models \varphi_{i_2}$ and $(\gamma_{i_1,i_2,m}, \rho_{i_1}) \models \varphi'_{i_2}$ such that $(\gamma_{i_1,i_2,0}, \gamma_{i_1,i_2,m}) \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k-1,m}$. As before, that happens iff there exist some $\gamma_{i_1,i_2,1}, \ldots, \gamma_{i_1,i_2,m-1} \in \mathcal{T}_{Cfg}$ (we already introduced $\gamma_{i_1,i_2,0}$ and $\gamma_{i_1,i_2,m}$) such that $(\gamma_{i_1,i_2,i_3}, \gamma_{i_1,i_2,i_3+1}) \in \mathcal{R}_{k-1,m}$ or $\gamma_{i_1,i_2,i_3} = \gamma_{i_1,i_2,i_3+1}$ for each $0 \le i_3 < m$. This procedure continues until k reaches 0, when no rules can be used, and thus only identical configuration pairs belong to $\bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{0,m}$ (as $\mathcal{R}^0_{0,m}$ is the identity).

During this process, the occurring configurations have indexes of the form $\gamma_{i_1,i_2,\ldots,i_{2j+1}}$ for some $0 \le j \le k$, with $i_1, i_3, \ldots, i_{2j-1}$ between 0 and $m-1$, with $i_2, i_4, \ldots, i_{2j}$ between 1 and nc, and with $0 \le i_{2j+1} \le m$. The intuition for such a configuration $\gamma_{i_1,\ldots,i_{2j+1}}$ is that it occurs when establishing a transition from position i_1 to i_1+1 on the path from γ to γ', and then when establishing a path for the i_2^{th} condition of the used rule for the said transition, and then when establishing a transition from position i_3 to position i_3+1 on the said path, and so on. Only the last index, i_{2j+1} can be m, as it can be either the source of a transition or the final configuration on a path. It is always the case that $(\gamma_{i_1,\ldots,i_{2j},0}, \gamma_{i_1,\ldots,i_{2j},m}) \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k-j,m}$.

With the above in mind, we introduce an indexing schema for configuration variables. For some $0 \le j \le k$, we use s to denote a sequence of indices $i_1, i_2, \ldots, i_{2j-1}, i_{2j}$ with each index i on an odd position in s ranging from 0 to $m-1$ and each i on an even position ranging from 1 to nc. Let $0 \le i \le m$. Then we use $c_{s,i}$ to denote a (fresh) variable of sort Cfg indexed by the sequence s, i. Intuitively, $c_{s,i}$ is interpreted as the configuration $\gamma_{i_1,\ldots,i_{2j},i_{2j+1}}$ mentioned above (with $i_{2j+1} = i$). Notice that c_0 and c_m (indexed by the sequences 0 and m, as s is empty) stand for γ and γ', the given configurations. Let I_j be the set of all such sequences s of length $2j$. Then we define the set of configuration variables associated with k and m as follows: $C_{k,m} = \{c_{s,i} \mid s \in I_j \text{ for some } 0 \le j \le k \text{ and } 0 \le i \le m\}$. Note that $C_{k,m}$ is finite. We quantify over finite sets of variables, like $\exists C_{k,m}$, as shorthand for quantifying over each variable in the set.

Let $\bar{x} = x_1, \ldots, x_{nx}$ be the free variables occurring in the rules in $\mathcal{S}$. Let $\mu \equiv (\varphi \Rightarrow \varphi' \text{ if } \varphi_1 \Rightarrow \varphi'_1 \wedge \ldots \wedge \varphi_n \Rightarrow \varphi'_n)$ be a rule in $\mathcal{S}$ and let s be a sequence of indices (as above) and $0 \le i \le nc$ be an index. Then we define the following FOL formula:

$$rule^{\mu}_{s,i} \equiv \exists\bar{x}\, (\varphi[c_{s,i}/\square] \wedge \varphi'[c_{s,i+1}/\square] \wedge \bigwedge_{1\le i'\le n} (\varphi_{i'}[c_{s,i,i',0}/\square] \wedge \varphi'_{i'}[c_{s,i,i',m}/\square])$$

Intuitively, $rule^{\mu}_{s,i}$ encodes the matching part of the definition of the transition relation $\rightarrow_{\mathcal{S}}$. It states that there exists some valuation of the free variables $x_1, \ldots, x_{nx}$ for which $c_{s,i}$ and $c_{s,i+1}$ match the LHS and RHS of μ, and $c_{s,i,i',0}$ and $c_{s,i,i',m}$ match the LHS and RHS of i'th condition of μ. Configuration variables are indexed by sequence s, i to avoid name conflicts. If μ is unconditional, then the big conjunction is empty. Now we can encode the existence of a path

$$path_{k,m} \equiv \bigwedge_{\substack{s \in I_k \\ 0 \le i < m}} c_{s,i} = c_{s,i+1} \wedge \bigwedge_{\substack{0 \le j < k \\ s \in I_j \\ 0 \le i < m}} (\bigvee_{\mu \in \mathcal{S}} rule^{\mu}_{s,i} \vee c_{s,i} = c_{s,i+1})$$

$$path(c, c') \equiv \exists k \exists m \exists C_{k,m} \, (path_{k,m} \wedge c_0 = c \wedge c_m = c')$$

The definition of $path(c, c')$ encodes $\rightarrow^{\star}_{\mathcal{S}}$ by encoding $\bigcup_{0 \le m' \le m} \mathcal{R}^{m'}_{k,m}$, and thus resembles the informal process above of establishing that $(\gamma, \gamma') \in \bigcup_{0 \le m' \le m} \mathcal{R}^{m'}_{k,m}$. The variables $c_{s,i}$ stand for the configurations $\gamma_{s,i}$. For each appropriate s and i, there is either a transition from $c_{s,i}$ to $c_{s,i+1}$ or $c_{s,i}$ is equal to $c_{s,i+1}$, unless s has length $2k$ when $c_{s,i}$ must be equal to $c_{s,i+1}$. Thus, there is some path from $c_{s,0}$ to $c_{s,m}$ of length at most m.

Using $path(c, c')$ we can encode the following: (1) the one step transition relation ($\rightarrow_{\mathcal{S}}$), (2) the termination dependence relation ($\succ$), (3) the divergence predicate ($\uparrow$), and (4) the configurations reaching some formula φ. For the definitions below, a rule μ is assumed of the form $\varphi \Rightarrow \varphi'$ if $\varphi_1 \Rightarrow \varphi'_1 \bigwedge \ldots \bigwedge \varphi_n \Rightarrow \varphi'_n$

$$\begin{aligned} step(c, c') \equiv\ & \exists c_1 \ldots c_{nc} \, \exists c'_1 \ldots c'_{nc} \exists \bar{x} \, (\bigvee_{\mu \in \mathcal{S}} (\varphi[c/\square] \wedge \varphi'[c'/\square] \\ & \wedge \bigwedge_{1 \le i \le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square] \wedge path(c_i, c'_i)))) \\ succ(c, c') \equiv\ & step(c, c') \\ & \vee \ \exists c_1 \ldots c_{nc} \, \exists c'_1 \ldots c'_{nc} \exists \bar{x} \, (\bigvee_{\mu \in \mathcal{S}} (\varphi[c/\square] \wedge \bigvee_{1 \le i \le n} (\varphi_i[c'/\square] \\ & \wedge \bigwedge_{1 \le j < i} (\varphi_j[c_j/\square] \wedge \varphi'_j[c'_j/\square] \wedge path(c_j, c'_j))))) \\ diverge(c) \equiv\ & \forall m \exists c_0 \ldots c_m (\bigwedge_{0 \le i < m} succ(c_i, c_{i+1}) \wedge c_0 = c) \\ coreach(\varphi) \equiv\ & \exists c \exists c' \, (c = \square \wedge \varphi[c'/\square] \wedge path(c, c')) \end{aligned}$$

These definitions are not (yet) proper FOL formulae: they quantify over sets and sequences of variables. The definitions of $\overline{path}(c, c')$ and $\overline{diverge}(c)$ in Figure 6.2 are proper FOL formulae equivalent to $path(c, c)$ and $diverge(c)$ (Lemma 22). Then the remaining predicates can also be expressed in FOL.

The following lemmas state various properties of the transition system, leading to the conclusion that the above definitions have the semantic properties their names suggest. First, we establish a FOL relation between $\mathcal{R}_{k+1,m}$ and $\bigcup_{0 \le m' \le m} \mathcal{R}^{m'}_{k,m}$:

Lemma 10. *Let $k, m \in \mathbb{N}$ and $c, c', c_1, c'_1, \ldots, c_{nc}, c'_{nc} \in Var_{Cfg}$ and $\gamma, \gamma' \in \mathcal{T}_{Cfg}$. Then $(\gamma, \gamma') \in \mathcal{R}_{k+1,m}$ iff there exists some $\rho : Var \rightarrow \mathcal{T}$ such that ($\mu \equiv \varphi \Rightarrow \varphi'$ if $\varphi_1 \Rightarrow \varphi'_1 \bigwedge \ldots \bigwedge \varphi_n \Rightarrow \varphi'_n$)*

$$\rho \models \bigvee_{\mu \in \mathcal{S}} (\varphi[c/\square] \wedge \varphi'[c'/\square] \wedge \bigwedge_{1 \le i \le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square]))$$

and $(\rho(c_i), \rho(c'_i)) \in \bigcup_{0 \le m' \le m} \mathcal{R}^{m'}_{k,m}$ for each $1 \le i \le nc$ and $\rho(c) = \gamma$ and $\rho(c') = \gamma'$. Moreover, $\gamma \rightarrow_{\mathcal{S}} \gamma'$ iff $\rho(c_i) \rightarrow^{\star}_{\mathcal{S}} \rho(c'_i)$ for each i.

Proof. For the direct implication, assume that $(\gamma, \gamma') \in \mathcal{R}_{k+1,m}$. Then there must be some rule $\mu \in \mathcal{S}$ and some ρ such that $(\gamma, \rho) \models \varphi$ and $(\gamma', \rho) \models \varphi'$ and for each $1 \le i \le n$ there exist γ_i, γ'_i with $(\gamma_i, \rho) \models \varphi_i$ and $(\gamma'_i, \rho) \models \varphi'_i$ such that $(\gamma_i, \gamma'_i) \in \bigcup_{0 \le m' \le m} \mathcal{R}^{m'}_{k,m}$. Since $c, c', c_1, c'_1, \ldots, c_{nc}, c'_{nc}$ do not occur in μ, we can assume that ρ is such that: $\rho(c) = \gamma$ and $\rho(c') = \gamma'$; and $\rho(c_i) = \gamma_i$ and $\rho(c'_i) = \gamma'_i$ for each $1 \le i \le n$; and $\rho(c_i) = \rho(c'_i)$ for each $n + 1 \le i \le nc$. Then we can conclude that $(\rho(c_i), \rho(c'_i)) \in \bigcup_{0 \le m' \le m} \mathcal{R}^{m'}_{k,m}$ for each $1 \le i \le nc$. By Lemma 7 we have that $(\rho(c), \rho) \models \varphi$ iff $\rho \models \varphi[c/\square]$ and $(\rho(c'), \rho) \models \varphi'$ iff $\rho \models \varphi'[c'/\square]$ and for each $1 \le i \le n$, $(\rho(c_i), \rho) \models \varphi_i$ iff $\rho \models \varphi_i[c_i/\square]$ and $(\rho(c'_i), \rho) \models \varphi'_i$ iff $\rho \models \varphi'_i[c'_i/\square]$. We can conclude that

$$\rho \models \varphi[c/\square] \wedge \varphi'[c'/\square] \wedge \bigwedge_{1 \le i \le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square])$$

and we are done.

For the reverse implication, we have that there must be some rule $\mu \in \mathcal{S}$ such that

$$\rho \models \varphi[c/\square] \wedge \varphi'[c'/\square] \wedge \bigwedge_{1 \le i \le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square])$$

Again, by Lemma 7 we have that $(\rho(c), \rho) \models \varphi$ iff $\rho \models \varphi[c/\square]$ and $(\rho(c'), \rho) \models \varphi'$ iff $\rho \models \varphi'[c'/\square]$ and for each $1 \le i \le n$, $(\rho(c_i), \rho) \models \varphi_i$ iff $\rho \models \varphi_i[c_i/\square]$ and $(\rho(c'_i), \rho) \models \varphi'_i$. Thus, it follows that $(\gamma, \rho) \models \varphi$ and $(\gamma', \rho) \models \varphi'$ and for each $1 \le i \le n$, $(\rho(c_i), \rho) \models \varphi_i$ and $(\rho(c_{i'}), \rho) \models \varphi'_i$. Therefore, $(\gamma, \gamma') \in \mathcal{R}_{k+1,m}$, and we are done.

We reduce the second part of the lemma to the first. For the direct implication, by Lemma 9, we have that $\gamma \rightarrow_{\mathcal{S}} \gamma'$ implies that there exist some k, m such that $(\gamma, \gamma') \in \mathcal{R}_{k+1,m}$. By the first part of the lemma, for each $1 \le i \le nc$, we have that $(\rho(c_i), \rho(c'_i)) \in \bigcup_{0 \le m' \le m} \mathcal{R}^{m'}_{k,m}$. Then, for each $1 \le i \le nc$, by Lemma 9, $(\rho(c_i), \rho(c'_i)) \in \bigcup_{0 \le m' \le m} \mathcal{R}^{m'}_{k,m}$ implies that $\rho(c_i) \rightarrow^{\star}_{\mathcal{S}} \rho(c'_i)$. For the converse implication, for each $1 \le i \le nc$, by Lemma 9, $\rho(c_i) \rightarrow^{\star}_{\mathcal{S}} \rho(c'_i)$ implies that there exist some k_i, m_i such that $(\rho(c_i), \rho(c'_i)) \in \bigcup_{0 \le m_i' \le m_i} \mathcal{R}^{m_i'}_{k_i,m_i}$. Let k be the maximum of k_i and m the maximum of m_i. By the first part of the lemma, we have that that $(\gamma, \gamma') \in \mathcal{R}_{k+1,m}$. By Lemma 9, we have that $(\gamma, \gamma') \in \mathcal{R}_{k+1,m}$ implies $\gamma \rightarrow_{\mathcal{S}} \gamma'$, and we are done. □

The following formally states the property encoded by $path_{k,m}$:

Lemma 11. *Let $k, m \in \mathbb{N}$. Then $(\gamma, \gamma') \in \bigcup_{0 \le m' \le m} \mathcal{R}^{m'}_{k,m}$ iff there exists a $\rho : Var \rightarrow \mathcal{T}$ such that $\rho \models path_{k,m}$ and $\rho(c_0) = \gamma$, $\rho(c_m) = \gamma'$.*

Proof. We proceed by induction on k.

Base case If $k = 0$, then I_k only contains the empty sequence and $path_{0,m}$ becomes $\bigwedge_{0 \le i < m} c_i = c_{i+1}$. Thus, $\rho \models path_{0,m}$ iff $\rho(c_0) = \ldots = \rho(c_m)$. Such a ρ exists iff $\gamma = \gamma'$. On the other hand, since $\mathcal{R}_{0,m} = \emptyset$, it follows that $\bigcup_{0 \le m' \le m} \mathcal{R}^{m'}_{0,m} = \mathcal{R}^{0}_{0,m}$, the reflexive closure of $\mathcal{R}_{0,m}$. Hence, we can conclude that $(\gamma, \gamma') \in \bigcup_{0 \le m' \le m} \mathcal{R}^{m'}_{0,m}$ iff $\gamma = \gamma'$, and we are done.

Induction case We assume the lemma for k and we prove it for $k + 1$. For $0 \le i_1 < m$ and $1 \le i_2 \le nc$ we define

$$\begin{aligned} path^{i_1,i_2}_{k,m} \equiv\ & \bigwedge_{\substack{s \in i_k \\ 0 \le i < m}} c_{i_1,i_2,s,i} = c_{i_1,i_2,s,i+1} \\ \wedge\ & \bigwedge_{\substack{0 \le j < k \\ s \in I_j \\ 0 \le i < m}} (\bigvee_{\mu \in \mathcal{S}} rule^{\mu}_{i_1,i_2,s,i} \vee c_{i_1,i_2,s,i} = c_{i_1,i_2,s,i+1}) \end{aligned}$$

Intuitively, $path^{i_1,i_2}_{k,m}$ is $path_{k,m}$ with all the indexing sequences prefixed with i_1 and i_2. Then we can rearrange $path_{k+1,m}$ as follows

$$\begin{aligned} path_{k+1,m} \leftrightarrow\ & \bigwedge_{0 \le i < m} (\bigvee_{\mu \in \mathcal{S}} rule^{\mu}_{i} \vee c_i = c_{i+1}) \\ \wedge\ & \bigwedge_{\substack{0 \le i_1 < m \\ 1 \le i_2 \le nc}} (\bigwedge_{\substack{s \in i_k \\ 0 \le i < m}} c_{i_1,i_2,s,i} = c_{i_1,i_2,s,i+1} \\ & \wedge \bigwedge_{\substack{0 \le j < k \\ s \in I_j \\ 0 \le i < m}} (\bigvee_{\mu \in \mathcal{S}} rule^{\mu}_{i_1,i_2,s,i} \vee c_{i_1,i_2,s,i} = c_{i_1,i_2,s,i+1})) \end{aligned}$$

Essentially, we split the conjuncts over s from $path_{k+1,m}$ based on whether s is empty (first line) or the first two elements are some i_1

and i_2 (second and third lines). Notice that for some fixed i_1 and i_2, the last two lines are in fact $path^{i_1,i_2}_{k,m}$. Thus we can write $path_{k+1,m}$ as

$$path_{k+1,m} \leftrightarrow \bigwedge_{0\le i<m} (\bigvee_{\mu\in\mathcal{S}} rule^{\mu}_i \vee c_i = c_{i+1}) \wedge \bigwedge_{\substack{0\le i_1<m\\ 1\le i_2\le nc}} path^{i_1,i_2}_{k,m}$$

Notice that, for given i_1 and i_2, the only variables (possibly) shared by $path^{i_1,i_2}_{k,m}$ with the rest of the formula are $c_{i_1,i_2,0}$ and $c_{i_1,i_2,m}$. Thus, the existence of ρ with $\rho \models path_{k+1,m}$ becomes equivalent to the existence of ρ' and ρ_{i_1,i_2} for each $0 \le i_1 < m$ and $1 \le i_2 \le nc$ with the following properties:

(1) $\rho' \models \bigwedge_{0\le i<m}(\bigvee_{\mu\in\mathcal{S}} rule^{\mu}_i \vee c_i = c_{i+1})$ and $\rho'(c_0) = \gamma$ and $\rho'(c_m) = \gamma'$; and
(2) $\rho_{i_1,i_2} \models path_{k,m}$ and $\rho_{i_1,i_2}(c_0) = \rho'(c_{i_1,i_2,0})$ and $\rho_{i_1,i_2}(c_m) = \rho'(c_{i_1,i_2,m})$ for each $0 \le i_1 < m$ and $1 \le i_2 \le nc$.

By the induction hypothesis, the existence of ρ_{i_1,i_2} satisfying condition (2) is equivalent to $(\rho'(c_{i_1,i_2,0}), \rho'(c_{i_1,i_2,m})) \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k,m}$. Thus, it suffices to prove that $(\gamma,\gamma') \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k+1,m}$ iff there exists some ρ' with $\rho' \models \bigwedge_{0\le i<m}(\bigvee_{\mu\in\mathcal{S}} rule^{\mu}_i \vee c_i = c_{i+1})$ such that $(\rho'(c_{i_1,i_2,0}), \rho'(c_{i_1,i_2,m})) \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k,m}$ for each $0 \le i_1 < m$ and $1 \le i_2 \le nc$.

Further, notice that for a given $0 \le i < m$, the only variables shared by $\bigvee_{\mu\in\mathcal{S}} rule^{\mu}_i \vee c_i = c_{i+1}$ with the rest of the formula are c_i and c_{i+1}. Thus, there existence of ρ' with the above properties is equivalent to the existence of some $\gamma_0,\ldots,\gamma_m \in \mathcal{T}_{Cfg}$ and some $\rho_0,\ldots,\rho_{m-1}$ such that for each $0 \le i < m$ it is the case that: $\rho_i(c_i) = \gamma_i$ and $\rho_i(c_{i+1}) = \gamma_{i+1}$; $(\rho_i(c_{i,i',0}), \rho_i(c_{i,i',m})) \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k,m}$, for each $1 \le i' \le nc$; and $\rho_i \models \bigvee_{\mu\in\mathcal{S}} rule^{\mu}_i \vee c_i = c_{i+1}$. By Lemma 10, we have that there exist some ρ_i with the first two properties such that $\rho_i \models \bigvee_{\mu\in\mathcal{S}} rule^{\mu}_i$ iff $(\gamma_i,\gamma_{i+1}) \in \mathcal{R}_{k+1,m}$. Thus, ρ_i exists iff $(\gamma_i,\gamma_{i+1}) \in \mathcal{R}_{k+1,m}$ or $\gamma_i = \gamma_{i+1}$. Therefore, it suffices to prove that $(\gamma,\gamma') \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k+1,m}$ iff there exist $\gamma_0,\ldots,\gamma_m$ such that for each $0 \le i < m$, either $(\gamma_i,\gamma_{i+1}) \in \mathcal{R}_{k+1,m}$ or $\gamma_i = \gamma_{i+1}$, which is trivial, and we are done. □

The following states that $path(c,c')$ encodes $\to^{\star}_{\mathcal{S}}$, the reflexive and transitive closure of the transition relation $\to_{\mathcal{S}}$:

Lemma 12. *Let $\rho : Var \to \mathcal{T}$. Then $\rho(c) \to^{\star}_{\mathcal{S}} \rho(c')$ iff $\rho \models path(c,c')$.*

Proof. We have that $\rho(c) \to^{\star}_{\mathcal{S}} \rho(c')$ iff there exist some $m' \in \mathbb{N}$ and some sequence $\gamma_0,\ldots,\gamma_{m'} \in \mathcal{T}_{Cfg}$ with $\gamma_0 = \rho(c)$ and $\gamma_{m'} = \rho(c')$ and $\gamma_i \to_{\mathcal{S}} \gamma_{i+1}$ for each $0 \le i < m'$. By Definition 5, $\gamma_i \to_{\mathcal{S}} \gamma_{i+1}$ iff for each $0 \le i \le m'$, there exists some k_i such that $(\gamma_i,\gamma_{i+1}) \in \mathcal{R}_{k_i}$. Further, by Lemma 9, that is iff for each $0 \le i \le m'$, there also exists some m_i such that $(\gamma_i,\gamma_{i+1}) \in \mathcal{R}_{k_i,m_i}$. Let k denote the maximum of $k_0,\ldots,k_{m-1}$ and m the maximum of $m',m_1,\ldots,m_{m'-1}$. Since $\mathcal{R}_{k_i,m_i} \subseteq \mathcal{R}_{k,m}$, we can conclude that $\rho(c) \to^{\star}_{\mathcal{S}} \rho(c')$ iff there exist some k and m such that $(\rho(c),\rho(c')) \in \bigcup_{0\le m'\le m} \mathcal{R}^{m'}_{k,m}$. By Lemma 11, that is iff there exists some ρ' with $\rho'(c_0) = \rho(c)$ and $\rho'(c_m) = \rho(c')$ such that $\rho' \models path_{k,m}$. Since c, c' do not occur in $path_{k,m}$, that is iff there exists some ρ' with $\rho'(c) = \rho(c)$ and $\rho'(c') = \rho(c')$ such that $\rho' \models path_{k,m} \wedge c_0 = c \wedge c_m = c'$. But since c, c' are the only free variables in $path(c,c')$, that holds iff $\rho \models path(c,c')$, and we are done. □

The following states that $step(c,c')$ encodes the transition relation $\to_{\mathcal{S}}$:

Lemma 13. *Let $\rho : Var \to \mathcal{T}$. Then $\rho(c) \to_{\mathcal{S}} \rho(c')$ iff $\rho \models step(c,c')$.*

Proof. By Lemma 10, second part, we have that $\rho(c) \to_{\mathcal{S}} \rho(c')$ iff there exists some ρ' with $\rho'(c) = \rho(c)$ and $\rho'(c') = \rho(c')$ and $\rho'(c_i) \to^{\star}_{\mathcal{S}} \rho'(c'_i)$ for each $1 \le i \le nc$ such that

$$\rho' \models \bigvee_{\mu\in\mathcal{S}} (\varphi[c/\square] \wedge \varphi'[c'/\square] \wedge \bigwedge_{1\le i\le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square]))$$

By Lemma 12, we have that $\rho'(c_i) \to^{\star}_{\mathcal{S}} \rho'(c'_i)$ iff $\rho' \models path(c_i,c'_i)$, for each $1 \le i \le nc$. Hence, we can combine the properties of ρ' into

$$\rho' \models \bigvee_{\mu\in\mathcal{S}} (\varphi[c/\square] \wedge \varphi'[c'/\square] \wedge \bigwedge_{1\le i\le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square])) \wedge \bigwedge_{1\le i\le nc} path(c_i,c'_i)$$

Then, by rearranging the above, we can conclude that $\rho(c) \to_{\mathcal{S}} \rho(c')$ iff there exist some ρ' with $\rho'(c) = \rho(c)$ and $\rho'(c') = \rho(c')$ such that

$$\rho' \models \bigvee_{\mu\in\mathcal{S}} (\varphi[c/\square] \wedge \varphi'[c'/\square] \wedge \bigwedge_{1\le i\le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square] \wedge path(c_i,c'_i)) \wedge \bigwedge_{n+1\le i\le nc} path(c_i,c'_i)) \tag{1}$$

Therefore, to prove the Lemma, it suffices to prove that there exists some ρ' with $\rho'(c) = \rho(c)$ and $\rho'(c') = \rho(c')$ satisfying (1) iff $\rho \models step(c,c')$.

For the direct implication, assume ρ' satisfy (1). Then, it follows that ρ' satisfies the first line in (1)

$$\rho' \models \bigvee_{\mu\in\mathcal{S}} (\varphi[c/\square] \wedge \varphi'[c'/\square] \wedge \bigwedge_{1\le i\le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square] \wedge path(c_i,c'_i)))$$

Since the free variables occurring in the formula above are among $c, c', c_1, c'_1, \ldots, c_{nc}, c'_{nc}$, then the existence of such a ρ' implies that

$$\rho \models \exists c_1 \ldots c_{nc}\ \exists c'_1 \ldots c'_{nc} \exists \bar{x}\ (\bigvee_{\mu\in\mathcal{S}} (\varphi[c/\square] \wedge \varphi'[c'/\square] \wedge \bigwedge_{1\le i\le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square] \wedge path(c_i,c'_i))))$$

which is exactly the definition of $step(c,c')$. Thus $\rho \models step(c,c')$, and we are done.

For the reverse implication, assume $\rho \models step(c,c')$. Then, by the definition of $step(c,c')$ and since c, c' are the only free variables $step(c,c')$, we have that there exists some ρ' with $\rho'(c) = \rho(c)$ and $\rho'(c') = \rho(c)$ such that

$$\rho' \models \bigvee_{\mu\in\mathcal{S}} (\varphi[c/\square] \wedge \varphi'[c'/\square] \wedge \bigwedge_{1\le i\le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square] \wedge path(c_i,c'_i)))$$

Then, there must be some $\mu \equiv \varphi \Rightarrow \varphi'$ if $\varphi_1 \Rightarrow \varphi'_1 \wedge \ldots \wedge \varphi_n \Rightarrow \varphi'_n$ such that

$$\rho' \models \varphi[c/\square] \wedge \varphi'[c'/\square] \wedge \bigwedge_{1\le i\le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square] \wedge path(c_i,c'_i))$$

Since the variables $c_{n+1}, c'_{n+1}, \ldots, c_{nc}, c'_{nc}$ do not occur in the formula above, we can assume that $\rho'(c_{n+1}) = \rho'(c'_{n+1}), \ldots, \rho'(c_{nc}) = \rho'(c'_{nc})$. Trivially, $\rho'(c_{n+1}) \to^{\star}_{\mathcal{S}} \rho'(c'_{n+1}), \ldots, \rho'(c_{nc}) \to^{\star}_{\mathcal{S}} \rho'(c'_{nc})$. By Lemma 12, it follows that $\rho' \models path(c_{n+1},c'_{n+1}), \ldots, \rho' \models path(c_{nc},c'_{nc})$. Thus, we can conclude that

$$\rho' \models \varphi[c/\square] \wedge \varphi'[c'/\square] \wedge \bigwedge_{1\le i\le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square] \wedge path(c_i,c'_i)) \wedge \bigwedge_{n+1\le i\le nc} path(c_i,c'_i)$$

and therefore that ρ' satisfies (1), and we are done. □

The following establishes a relation between $path(c,c')$ and $step(c,c')$ which we use later on in Section 6.4:

Lemma 14. $\models path(c,c') \leftrightarrow c = c' \vee \exists c''\ (step(c,c'') \wedge path(c'',c'))$.

Proof. We prove that $\rho \models c = c' \vee \exists c''\ (\mathit{step}(c,c'') \wedge \mathit{path}(c'',c'))$ iff $\rho \models \mathit{path}(c,c')$. By Lemma 12, $\rho \models \mathit{path}(c,c')$ iff $\rho(c) \rightarrow^{\star}_{\mathcal{S}} \rho(c')$, that is, iff there exist some $\gamma_0, \ldots, \gamma_n$ with $\rho(c) = \gamma_0$ and $\rho(c) = \gamma_n$ and $\gamma_i \rightarrow_{\mathcal{S}} \gamma_{i+1}$ for each $0 \leq i < n$. Equivalently, we can state it as either $\rho(c) = \rho(c')$ or there exist some $\gamma_0, \gamma_1, \gamma_n$ with $\rho(c) = \gamma_0$ and $\rho(c) = \gamma_n$ such that $\gamma_0 \rightarrow_{\mathcal{S}} \gamma_1$ and $\gamma_1 \rightarrow^{\star}_{\mathcal{S}} \gamma_n$. Further, that is iff there exists some ρ' with $\rho'(c) = \rho(c)$ and $\rho'(c') = \rho(c')$ such that $\rho'(c) \rightarrow_{\mathcal{S}} \rho'(c'')$ and $\rho'(c'') \rightarrow^{\star}_{\mathcal{S}} \rho'(c')$. By Lemma 13 and Lemma 12, that is iff $\rho' \models \mathit{step}(c,c'') \wedge \mathit{path}(c'',c')$, and we are done. □

The following states that $\mathit{succ}(c,c')$ encodes the termination dependence relation $\succ$:

Lemma 15. *Let $\rho : \mathit{Var} \rightarrow \mathcal{T}$. Then $\rho(c') \prec \rho(c)$ iff $\rho \models \mathit{succ}(c,c')$.*

Proof. Recall from Definition 7 that $\gamma \succ \gamma'$ iff

- $\gamma \rightarrow_{\mathcal{S}} \gamma'$; or
- there exists some rule
$$\varphi \Rightarrow \varphi' \text{ if } \varphi_1 \Rightarrow \varphi'_1 \wedge \ldots \wedge \varphi_n \Rightarrow \varphi'_n$$
in $\mathcal{S}$, valuation $\rho' : \mathit{Var} \rightarrow \mathcal{T}$, and index $1 \leq i \leq n$ such that:
  (1) $(\gamma, \rho') \models \varphi$;
  (2) for each $1 \leq j < i$ and each $\gamma_j \in \mathcal{T}_{\mathit{Cfg}}$ with $(\gamma_j, \rho') \models \varphi_j$, there is some $\gamma'_j \in \mathcal{T}_{\mathit{Cfg}}$ such that $(\gamma'_j, \rho') \models \varphi'_j$ and $\gamma_j \rightarrow^{\star}_{\mathcal{S}} \gamma'_j$; and
  (3) $(\gamma', \rho') \models \varphi_i$.

For each $1 \leq j < i$, since φ_j is well-defined (see Definition 9), there exists a unique γ_j with $(\gamma_j, \rho') \models \varphi_j$. Hence, condition (2) is equivalent to: "for each $1 \leq j < i$ there exist some $\gamma_j \in \mathcal{T}_{\mathit{Cfg}}$ with $(\gamma_j, \rho') \models \varphi_j$ and some $\gamma'_j \in \mathcal{T}_{\mathit{Cfg}}$ such that $(\gamma'_j, \rho') \models \varphi'_j$ and $\gamma_j \rightarrow^{\star}_{\mathcal{S}} \gamma'_j$". By Lemma 13, $\rho(c) \rightarrow_{\mathcal{S}} \rho(c')$ iff $\mathit{step}(c,c')$. Thus, the first line in the definition of $\mathit{succ}(c,c')$ captures the first bullet in the definition of $\succ$. Therefore, it suffice to show that the second and third lines of the definition of $\mathit{succ}(c,c')$, namely

$$\rho \models \exists c_1 \ldots c_{nc}\, \exists c'_1 \ldots c'_{nc} \exists \bar{x}\ (\bigvee_{\mu \in \mathcal{S}} (\varphi[c/\square] \wedge \bigvee_{1 \leq i \leq n} (\varphi_i[c'/\square] \wedge \bigwedge_{1 \leq j < i} (\varphi_j[c_j/\square] \wedge \varphi'_j[c'_j/\square] \wedge \mathit{path}(c_j, c'_j)))))$$

capture the second bullet. Since c, c' are the only free variables in the formula above, it follows that ρ satisfies it iff there exists some ρ' with $\rho'(c) = \rho(c)$ and $\rho'(c') = \rho(c')$ such that

$$\rho' \models \bigvee_{\mu \in \mathcal{S}} (\varphi[c/\square] \wedge \bigvee_{1 \leq i \leq n} (\varphi_i[c'/\square] \wedge \bigwedge_{1 \leq j < i} (\varphi_j[c_j/\square] \wedge \varphi'_j[c'_j/\square] \wedge \mathit{path}(c_j, c'_j))))$$

or, equivalently, iff there exist some rule μ and $1 \leq i \leq n$ such that

$$\rho' \models \varphi[c/\square] \wedge \varphi_i[c'/\square] \bigwedge_{1 \leq j < i} (\varphi_j[c_j/\square] \wedge \varphi'_j[c'_j/\square] \wedge \mathit{path}(c_j, c'_j))$$

By Lemma 7, we have that $\rho' \models \varphi[c/\square]$ iff $(\rho'(c), \rho') \models \varphi$ and $\rho' \models \varphi_i[c'/\square]$ iff $(\rho'(c'), \rho') \models \varphi_i$, which are condition (1) and (3). Also by Lemma 7, for each $1 \leq j < i$, we have that $\rho' \models \varphi_j[c_j/\square]$ iff $(\rho'(c_j), \rho') \models \varphi_j$ and $\rho' \models \varphi'_j[c'_j/\square]$ iff $(\rho'(c'_j), \rho') \models \varphi'_j$, and, by Lemma 12, that $\rho' \models \mathit{path}(c_j, c'_j)$ iff $\rho'(c_j) \rightarrow^{\star}_{\mathcal{S}} \rho'(c'_j)$, which is condition (2). We can conclude that the existence of some ρ' satisfying the formula above is equivalent to the existence of some ρ' satisfying the second bullet, and we are done. □

The following states that $\mathit{diverge}(c)$ encodes the divergence predicate $\uparrow$:

Lemma 16. *Let $\rho : \mathit{Var} \rightarrow \mathcal{T}$. Then $\rho \models \mathit{diverge}(c)$ iff $\rho(c)$ does not terminate.*

Proof. For an arbitrary γ, we let $\mathit{Prop}(\gamma)$ be the property stating that there exists an infinite set P_γ of finite $\succ$-sequences starting at γ. First we prove that $\mathit{Prop}(\gamma)$ holds iff γ does not terminate. For the direct implication, we inductively construct an infinite $\succ$-sequence $\gamma_0, \ldots, \gamma_n, \ldots$ such that $\gamma_0 = \gamma$ and $\mathit{Prop}(\gamma_n)$ holds for all n. $\mathit{Prop}(\gamma_0)$ holds because $\gamma_0 = \gamma$. Now, let us inductively assume $\mathit{Prop}(\gamma_n)$ holds and let $\succ(\gamma_n) = \{\gamma \mid \gamma_n \succ \gamma\}$ be the set of successors of γ_n. For each $\gamma \in \succ(\gamma_n)$, let P'_γ be the set $\{\tau \mid \tau \in P_{\gamma_n}$ and $\gamma_n\gamma$ is a prefix of $\tau\}$. Clearly, the sets P'_γ form a partition of P_{γ_n}. One of the assumption of relative completeness is that each configuration has a finite number of $\succ$-successors, that is, $\succ(\gamma_n)$ is finite. Since P_{γ_n} is infinite (because we assumed $\mathit{Prop}(\gamma_n)$), there is at least one $\gamma \in \succ (\gamma_n)$ with P'_γ infinite. Then we choose γ_{n+1} to be γ. Note that $\gamma_n \succ \gamma_{n+1}$ and $P_{\gamma_{n+1}} = \{\tau \mid \gamma_n\tau \in P'_{\gamma_{n+1}}\}$ is infinite, thus $\mathit{Prop}(\gamma_{n+1})$ holds. We can conclude that γ does not terminate. For the converse implication, it suffices to notice that if there is an infinite $\succ$-sequence starting at γ, then the set of finite prefixes of that sequence is infinite. A direct consequence of this result is that γ does not terminate iff for each n, there exists a $\succ$-sequence of length n starting at γ. Indeed, if γ does not terminate, then for each n we can take the prefix of length n of the infinite sequence starting at γ. Conversely, if there exists some $\succ$-sequence starting at γ for each n, then the set of such sequences if infinite, and thus γ does not terminate.

Using the result above, it suffices to prove that $\rho \models \mathit{diverge}(c)$ iff for each m there exists some $\succ$-sequence starting at $\rho(c)$. By Lemma 15, we have that for each m there exists some ρ' with $\rho'(c) = \rho(c)$ such that $\rho' \models \bigwedge_{0 \leq i < m} \mathit{succ}(c_i, c_{i+1}) \wedge c_0 = c$ iff $\rho'(c_0) = \rho'(c) = \rho(c)$ and $\rho'(c_0) \succ \ldots \succ \rho'(c_m)$, that is, iff there is some $\succ$-sequence of length m starting at $\rho(c)$, and we are done. □

The following establishes a relation between $\mathit{diverge}(c)$ and $\mathit{succ}(c,c')$ which we use later on in Section 6.4:

Lemma 17. $\models \mathit{diverge}(c) \leftrightarrow \exists c'\ (\mathit{succ}(c,c') \wedge \mathit{diverge}(c'))$.

Proof. We prove that $\rho \models \mathit{diverge}(c)$ iff $\rho \models \exists c'\ (\mathit{succ}(c,c') \wedge \mathit{diverge}(c'))$. By Lemma 16, $\rho \models \mathit{diverge}(c)$ iff $\rho(c)$ does not terminate, that is, iff there exist some $\gamma_0, \ldots, \gamma_n, \ldots$ with $\rho(c) = \gamma_0$ and $\gamma_i \succ \gamma_{i+1}$ for each $i \geq 0$. Equivalently, we can state it as there exist some γ_0, γ_1 with $\rho(c) = \gamma_0$ such that $\gamma_0 \succ \gamma_1$ and γ_1 does not terminate. Further, that is iff there exists some ρ' with $\rho'(c) = \rho(c)$ such that $\rho'(c) \succ \rho'(c')$ and $\rho'(c')$ does not terminate. By Lemma 16, that is iff $\rho' \models \mathit{succ}(c,c') \wedge \mathit{diverge}(c)$, and we are done. □

The following summarises the main properties of $\mathit{step}(c,c')$, $\mathit{path}(c,c')$, $\mathit{succ}(c,c')$, and $\mathit{diverge}(c)$:

Lemma 18. *Let $\rho : \mathit{Var} \rightarrow \mathcal{T}$. Then $\rho(c) \rightarrow_{\mathcal{S}} \rho(c')$ iff $\rho \models \mathit{step}(c,c')$; $\rho(c) \rightarrow^{\star}_{\mathcal{S}} \rho(c')$ iff $\rho \models \mathit{path}(c,c')$; $\rho(c') \prec \rho(c)$ iff $\rho \models \mathit{succ}(c,c')$; and $\rho(c)$ does not terminate iff $\rho \models \mathit{diverge}(c)$.*

Proof. Follows by Lemmas 13 , 12 , 15 and 16. □

Finally, the following establishes the property of $\mathit{coreach}(\varphi)$:

Lemma 19. *Let $\gamma \in \mathcal{T}_{\mathit{Cfg}}$ and $\rho : \mathit{Var} \rightarrow \mathcal{T}$. Then $(\gamma, \rho) \models \mathit{coreach}(\varphi)$ iff there exists some $\gamma' \in \mathcal{T}_{\mathit{Cfg}}$ with $(\gamma', \rho) \models \varphi$ and $\gamma \rightarrow^{\star}_{\mathcal{S}} \gamma'$.*

Proof. We have that $(\gamma, \rho) \models \mathit{coreach}(\varphi)$ iff there exists some ρ' which agrees with ρ on $\mathit{Var} \setminus \{c, c'\}$ with $\rho'(c) = \gamma$ such that $\rho' \models \varphi[c'/\square] \wedge \mathit{path}(c,c')$. By Lemma 12, that is iff there exists some ρ' which agrees with ρ on $\mathit{Var} \setminus \{c, c'\}$ with $\rho'(c) = \gamma$ such that $\rho' \models \varphi[c'/\square]$ and $\gamma \rightarrow^{\star}_{\mathcal{S}} \rho'(c')$. Let us denote $\rho(c')$ by γ'. Then $(\gamma, \rho) \models \mathit{coreach}(\varphi)$ iff there exist some γ' and some ρ' which agrees with ρ on $\mathit{Var} \setminus \{c, c'\}$ with $\rho'(c) = \gamma$ and $\rho'(c') = \gamma'$ such that $\rho' \models \varphi[c'/\square]$ and $\gamma \rightarrow^{\star}_{\mathcal{S}} \gamma'$. We have that $(\rho'(c'), \rho') \models \varphi$ iff $\rho' \models \varphi[c'/\square]$. Since c, c' do not occur in φ, and ρ and ρ' agree on $\mathit{Var} \setminus \{c, c'\}$, it follows that $\rho' \models \varphi[c'/\square]$ iff $(\gamma', \rho) \models \varphi$. Thus, we can conclude that $(\gamma, \rho) \models \mathit{coreach}(\varphi)$ iff there exists some γ' with $(\gamma', \rho) \models \varphi$ and $\gamma \rightarrow^{\star}_{\mathcal{S}} \gamma'$, and we are done. □

$$
\begin{aligned}
\overline{rule}_\mu(\bar{s}, j, i) \equiv{}& \exists c \exists c' \exists c_1 \ldots c_n \exists c'_1 \ldots c'_n \; (\exists \bar{x} \; (\varphi[c/\Box] \wedge \varphi'[c'/\Box] \wedge \bigwedge_{1 \le i' \le n} \varphi_{i'}[c_{i'}/\Box] \wedge \varphi'_{i'}[c'_{i'}/\Box]) \wedge \beta(a, b, add(\bar{s}, j, i), \alpha(c)) \\
& \wedge \beta(a, b, add(\bar{s}, j, i+1), \alpha(c')) \wedge \bigwedge_{1 \le i' \le n} (\beta(a, b, add(\bar{s}, j, i, i', 0), \alpha(c_{i'})) \wedge \beta(a, b, add(\bar{s}, j, i, i', m), \alpha(c'_{i'})))) \\
\overline{id}(\bar{s}, j, i) \equiv{}& \exists c \exists c' \; (\beta(a, b, add(\bar{s}, j, i), \alpha(c)) \wedge \beta(a, b, add(\bar{s}, j, i+1), \alpha(c')) \wedge c = c') \\
\overline{path}(c, c') \equiv{}& \exists k \exists m \; (k \ge 0 \wedge m \ge 0 \wedge \exists a \exists b \; (\exists c_0 \; (\beta(a, b, 0, \alpha(c_0)) \wedge c = c_0) \wedge \exists c_m \; (\beta(a, b, m, \alpha(c_m)) \wedge c' = c_m) \\
& \wedge \forall \bar{s} \forall j \forall i \; (seq(\bar{s}, j) \wedge j = k \wedge 0 \le i < m \to \overline{id}(\bar{s}, j, i)) \wedge \forall \bar{s} \forall j \forall i \; (seq(\bar{s}, j) \wedge j < k \wedge 0 \le i < m \to \bigvee_{\mu \in \mathcal{S}} \overline{rule}_\mu(\bar{s}, j, i) \vee \overline{id}(\bar{s}, j, i)))) \\
\overline{diverge}(c) \equiv{}& \forall m \exists a \exists b \; (\exists c_0 \; (\beta(a, b, 0, \alpha(c_0)) \wedge c = c_0) \wedge \forall i \; (0 \le i < m \to \exists c_i \exists c_{i+1} \; (\beta(a, b, i, \alpha(c_i)) \wedge \beta(a, b, i+1, \alpha(c_{i+1})) \wedge succ(c_i, c_{i+1}))))
\end{aligned}
$$

Figure 4. FOL definitions of a finite $\to_{\mathcal{S}}$-sequence ($\overline{path}$) and an infinite $\succ$-sequence ($\overline{diverge}$)

6.3 Formulae Gödelization

We use Gödel's β predicate to encode quantification over sequences of configurations in FOL (see [37] for an accessible introduction to Gödelization and the β predicate). The predicate β relies on the reminder of a when divided by b, written $a \bmod b$ and defined as

$$r = a \bmod b \;\; \equiv \;\; \exists d \; (b \times d \le a \wedge b \times (d+1) > a \wedge a = b \times d + r)$$

Gödel's $\beta(a, \; b, \; i, \; x)$ predicate over natural numbers is defined as

$$\beta(a, \; b, \; i, \; x) \;\; \equiv \;\; x = a \bmod (1 + (1+i) \times b)$$

The assumptions on the model $\mathcal{T}$ allow us to express β. If $u_0, \ldots, u_n$ is a sequence of natural numbers, then there exist natural numbers a and b such that $\beta(a, b, i, x)$ holds iff $x = u_i$, for each $0 \le i \le n$ and x. Thus, for any given n, we can systematically translate sentences $\exists u_0, \ldots, u_n \; \varphi$ into equivalent sentences $\exists a, b \, \overline{\varphi}$. As part of this translation, each atomic formula $pred$ of φ is translated into

$$\exists u_{i_1}, \ldots, u_{i_k} \; (\beta(a, \; b, \; i_1, \; u_{i_1}) \wedge \cdots \wedge \beta(a, \; b, \; i_k, \; u_{i_k}) \wedge pred)$$

where $u_{i_1}, \ldots, u_{i_k}$ are all the variables among $u_0, \ldots, u_n$ occurring in $pred$. Even if n itself is quantified, only a fixed (independent of n) subset of the variables $u_0, \ldots, u_n$ can occur in $pred$, making $\overline{\varphi}$ a proper FOL formula. Thus, β enables quantification over sequences.

Using the injective function $\alpha : \mathcal{T}_{Cfg} \to \mathbb{N}$ we can extend the result above to sequences of configurations. Sentences of the form $\exists c_0, \ldots, c_n \; \varphi$ can be systematically translated into equivalent sentences of the form $\exists a, b \, \overline{\varphi}$, where $\overline{\varphi}$ is a FOL formula replacing each atomic formula $pred$ of φ containing variables $c_{i_1}, \ldots, c_{i_k}$ with

$$\exists c_{i_1}, \ldots, c_{i_k} \; (\beta(a, \; b, \; i_1, \; \alpha(c_{i_1})) \wedge \cdots \wedge \beta(a, \; b, \; i_k, \; \alpha(c_{i_k})) \wedge pred)$$

The injectivity of α guarantees that different free occurrences of the same variable c_i in φ are correctly related in $\overline{\varphi}$.

Thus far, we can encode quantification over finite sequences of configurations. A finite sequences is (syntactically) represented as a finite set of variables indexed by natural numbers. However, $path(c, c')$ quantifies over the set $C_{k,m}$, which contain variables indexed by sequences s, i. To encode a such a sequence s, i into a natural number, we need division ($a \div b$) and power (a^b), which can be defined as follows.

$$
\begin{aligned}
a \div b = d &\equiv b \times d \le a \wedge b \times (d+1) > a \\
a^b = d &\equiv \exists x_0 \ldots x_b \; (x_0 = 1 \wedge x_b = d \wedge \forall i \; (1 \le i \le b \to x_i = x_{i-1} \times a))
\end{aligned}
$$

Equivalently, we can define a^b using β and only four quantifiers.

$$
\begin{aligned}
a^b = d \;\; \equiv \;\; & b \ge 0 \wedge \exists a' \exists b' \; (\beta(a', b', 0, 1) \wedge \beta(a', b', b, d) \wedge \forall i \; (1 \le i \le b \\
& \to \exists x \exists x' \; (\beta(a', b', i, x)) \wedge \beta(a', b', i-1, x') \wedge x = x' \times a)))
\end{aligned}
$$

For notational simplicity, we write division, reminder, and power as functions rather than as relations. Also, to save space, we write $a \le b \le d$ instead of $a \le b \wedge b \le d$. Let p be the maximum of m and $nc + 1$, and let $s = i_1, \ldots, i_{2j}$ be a sequence of indices. Recall that $i_1, i_3, \ldots i_{2j-1}$ are between 0 and $m-1$, and $i_2, i_4, \ldots, i_{2j}$ between 1 and nc. Then, we can view s as a number $\bar{s}$ in base p, namely $\bar{s} = \Sigma_{t=1}^{2j} i_t \times p^{t-1}$. Conversely, we encode the fact that $\bar{s}$ is indeed representing some sequence s as follows

$$
\begin{aligned}
seq(\bar{s}, j) \equiv{}& \bar{s} = 0 \wedge j = 0 \\
& \vee \; j > 0 \wedge p^{2 \times j - 1} \le \bar{s} < p^{2 \times j} \\
& \quad \wedge \forall t \; (1 \le t \le j \to \exists d \; (d = (\bar{s} \div p^{2 \times t - 2}) \bmod p \wedge 0 \le d < m)) \\
& \quad \wedge \forall t \; (1 \le t \le j \to \exists d \; (d = (\bar{s} \div p^{2 \times t - 1}) \bmod p \wedge 1 \le d \le nc))
\end{aligned}
$$

Intuitively, the first line covers the case of the empty sequence ($j = 0$). For the case of non-empty sequences, the second line states that $\bar{s}$ has exactly $2j$ digits, the third that the digits on even positions (corresponding to $i_1, \ldots, i_{2j-1}$) are between 0 and $m-1$, and the fourth that the digits on odd positions (corresponding to $i_2, \ldots, i_{2j}$) are between 1 and nc. Similarly, to the sequence s, i we associate the number $\Sigma_{t=1}^{2j} i_t \times p^{t-1} + i \times p^{2 \times j}$. For convenience, we define

$$
\begin{aligned}
add(\bar{s}, j, a) &\equiv \bar{s} + a \times p^{2 \times j} \\
add(\bar{s}, j, a, b, d) &\equiv \bar{s} + a \times p^{2 \times j} + b \times p^{2 \times j + 1} + d \times p^{2 \times j + 2}
\end{aligned}
$$

For some $\bar{s}$ associated to some s of length $2j$, these functions give the numbers associated to the sequences s, a and s, a, b, d.

Figure 6.2 presents the encoding of a finite $\to_{\mathcal{S}}$-sequence ($\overline{path}(c, c')$) and an infinite $\succ$-sequence ($\overline{diverge}(c)$) using only a fixed number of quantifiers. Recall that $path(c, c')$ existentially quantifies over the finite set of variables $C_{k,m}$. Using the encoding of sequences s, i into numbers in base p with at most $2k + 1$ digits, we can instead quantify over a sequence of configurations of length at most p^{2k+1}. Further, using the β predicate and the injective function $\alpha : \mathcal{T}_{Cfg} \to \mathbb{N}$, we can instead quantify over two natural numbers, namely a, b. Then, we can replace the big conjunctions in $path(c, c')$ with universal quantification over $\bar{s}, j, i$ and the appropriate restrictions like $seq(\bar{s}, j)$, $0 \le i < m$, etc. We introduce $\overline{rule}_\mu(\bar{s}, j, i)$ as the encoding of $rule^\mu_{s,i}$. It replaces the configuration variables indexed by sequences with locally quantified variables $c, c', c_1, c'_1, \ldots, c_n, c'_n$ (n is the number of conditions of μ), and it existentially quantifies $\bar{x}$, the variables occurring free in the rules. It also ensures that these local configuration variables are instantiated consistently across the formula. For example, for the variable $c_{s,i}$, the predicate $\beta(a, \; b, \; add(\bar{s}, j, i), \; \alpha(c))$ ensures that the variable c is instantiated with the value intended for $c_{s,i}$, that is, the configuration mapped by α into the number on the position $\bar{s} + i \times p^{2 \times j}$ of the sequence of natural numbers Gödelized by a and b. Similarly, $\overline{id}(\bar{s}, j, i)$ encodes $c_{s,i} = c_{s,i+1}$. Formally, we have the following result:

Lemma 20. $\models path(c, c') \leftrightarrow \overline{path}(c, c')$.

Proof. Let $\rho, \bar{\rho} : Var \to \mathcal{T}$. We call $\rho, \bar{\rho}$ a *Gödel pair* iff

(1) ρ and $\bar{\rho}$ agree on c, c', k, m

(2) for each indexing sequence s of length $2j$ with $0 \le j \le \rho(k)$ and for each $0 \le i \le \rho(m)$ it is the case that

$$\beta(\bar{\rho}(a), \; \bar{\rho}(b), \; \bar{s} + i \times p^{2 \times j}, \; \alpha(\rho(c_{s,i})))$$

holds, where p is the maximum of $\rho(m)$ and nc. Intuitively, this states that the number associated to the configuration $\rho(c_{s,i})$ by the injective function α is on the position $\bar{s} + i \times p^{2 \times j}$ (the position associated to the sequence s, i) in the sequence of natural numbers Gödelized by the pair $\bar{\rho}(a), \bar{\rho}(b)$.

We intend to use ρ to interpret the top-level existential quantifiers in $path(c, c')$ and $\bar{\rho}$ to interpret the top-level existential quantifiers in $\overline{path}(c, c')$. To simplify the notation, for variables a of sort $\mathbb{N}$, like k, m, a, b, i, j, etc, we use its syntactic name, e.g. k, to also refer to its interpretation, e.g. $\rho(k)$. Condition (1) above ensures that variables common to both $path(c, c')$ and $\overline{path}(c, c')$ are interpreted consistently by ρ and $\bar{\rho}$, and by valuations which agree with ρ or $\bar{\rho}$ on these common variables. With this convention, the instance of the β predicate above becomes

$$\beta(a,\ b,\ \bar{s} + i \times p^{2\times j},\ \alpha(\rho(c_{s,i})))$$

We prove that for each Gödel pair $\rho, \bar{\rho}$, the following two statements are equivalent

$$\rho \models c = c_0 \wedge c' = c_m \wedge path_{k,m} \tag{3}$$

$$\begin{aligned}
\bar{\rho} \models\ & \exists c_0\ (\beta(a, b, 0, \alpha(c_0)) \wedge c = c_0) \\
& \wedge \exists c_m\ (\beta(a, b, m, \alpha(c_m)) \wedge c = c_m) \\
& \wedge \forall \bar{s} \forall j \forall i\ (seq(\bar{s}, j) \wedge j = k \wedge 0 \le i < m \rightarrow \overline{id}(\bar{s}, j, i)) \\
& \wedge \forall \bar{s} \forall j \forall i\ (seq(\bar{s}, j) \wedge j < k \wedge 0 \le i < m \\
& \qquad \rightarrow \bigvee_{\mu \in \mathcal{S}} \overline{rule}_\mu(\bar{s}, j, i) \vee \overline{id}(\bar{s}, j, i))
\end{aligned} \tag{4}$$

First, we prove that $\rho \models rule^\mu_{s,i}$ iff $\bar{\rho} \models \overline{rule}_\mu(\bar{s}, j, i)$ for each indexing sequence s of length $2j$ with $0 \le j < k$ and each $0 \le i \le m$ and each $\bar{s}$ such that $\bar{s}$ is the number associated to the sequence s, i. Recall that $\overline{rule}_\mu(\bar{s}, j, i)$ is obtained from $rule^\mu_{s,i}$ by

- substituting $c_{s,i}, c_{s,i+1}$ with c, c' and $c_{s,i,i',0}, c_{s,i,i',m}$ with c_j, c'_j for each $1 \le i' \le n$;
- by adding constrains on $c, c', c_1, c'_1, \ldots, c_n, c_n$ using the β predicate and the α function; and
- existentially quantifying $c, c', c_1, c'_1, \ldots, c_n, c'_n$.

Let $\bar{\rho}_\mu : Var \rightarrow \mathcal{T}$ which agrees with $\bar{\rho}$ on $Var \setminus \{c, c', c_1, c'_1, \ldots, c_n, c'_n\}$. Then we have that $\bar{\rho}_\mu$ satisfies the constrains on the configuration variables in the definition of $\overline{rule}_\mu(\bar{s}, j, i)$, namely

$$\begin{aligned}
\rho_\mu \models\ & \beta(a, b, add(\bar{s}, j, i), \alpha(c)) \wedge \beta(a, b, add(\bar{s}, j, i+1), \alpha(c')) \\
& \wedge \bigwedge_{1 \le i' \le n} (\beta(a, b, add(\bar{s}, j, i, i', 0), \alpha(c_{i'})) \\
& \qquad \wedge \beta(a, b, add(\bar{s}, j, i, i', m), \alpha(c'_{i'})))
\end{aligned}$$

iff $\beta(a, b, add(\bar{s}, j, i), \alpha(\bar{\rho}_\mu(c)))$ and $\beta(a, b, add(\bar{s}, j, i+1), \alpha(\bar{\rho}_\mu(c')))$ hold, and for each $1 \le i' \le n$, $\beta(a, b, add(\bar{s}, j, i, i', 0), \alpha(\bar{\rho}_\mu(c_{i'})))$ and $\beta(a, b, add(\bar{s}, j, i, i', m), \alpha(\bar{\rho}_\mu(c'_{i'})))$ hold. Since ρ and $\bar{\rho}$ are a Gödel pair, condition (2) implies that $\beta(a, b, add(\bar{s}, j, i), \alpha(\rho(c_{s,i})))$ and $\beta(a, b, add(\bar{s}, j, i+1), \alpha(\rho(c_{s,i+1})))$ hold, and also that for each $1 \le i' \le n$, it is the case that $\beta(a, b, add(\bar{s}, j, i, i', 0), \alpha(\rho(c_{s,i,i',0})))$ and $\beta(a, b, add(\bar{s}, j, i, i', m), \alpha(\rho(c'_{s,i,i',m})))$ hold. Further, since α is injective, and the first three arguments of β uniquely determine the fourth argument, we can conclude there exists a unique $\bar{\rho}_\mu$ which satisfies the constrains above, namely the one with $\bar{\rho}_\mu(c) = \rho(c_{s,i})$ and $\bar{\rho}_\mu(c') = \rho(c_{s,i+1})$, and with $\bar{\rho}_\mu(c_{i'}) = \rho(c_{s,i,i',0})$ and $\bar{\rho}_\mu(c'_{i'}) = \rho(c_{s,i,i',m})$ for each $1 \le i' \le n$. Then, it follows that $\rho \models rule^\mu_{s,i}$ iff $\bar{\rho}_\mu$ satisfies $rule^\mu_{s,i}$ with the configuration variables substituted, that is, iff $\bar{\rho} \models \overline{rule}_\mu(\bar{s}, j, i)$.

Similarly, we prove that $\rho \models c_{s,i} = c_{s,i+1}$ iff $\bar{\rho} \models \overline{id}(s, j, i)$ for each indexing sequence s of length $2j$ with $0 \le j \le k$ and each $0 \le i \le m$ and each $\bar{s}$ such that $\bar{s}$ is the number associated to the sequence s, i. Let $\bar{\rho}_{\mathrm{id}} : Var \rightarrow \mathcal{T}$ which agrees with $\bar{\rho}$ on $Var \setminus \{c, c'\}$. Then we have that $\bar{\rho}_{\mathrm{id}} \models \beta(a, b, add(\bar{s}, j, i), \alpha(c)) \wedge \beta(a, b, add(\bar{s}, j, i+1), \alpha(c'))$ iff $\beta(a, b, add(\bar{s}, j, i), \alpha(\bar{\rho}_{\mathrm{id}}(c)))$ and $\beta(a, b, add(\bar{s}, j, i+1), \alpha(\bar{\rho}_{\mathrm{id}}(c')))$ hold. Since ρ and $\bar{\rho}$ are a Gödel pair, condition (2) implies that $\beta(a, b, add(\bar{s}, j, i), \alpha(\rho(c_{s,i})))$ and $\beta(a, b, add(\bar{s}, j, i+1), \alpha(\rho(c_{s,i+1})))$ hold. Then, we can conclude there exists a unique $\bar{\rho}_{\mathrm{id}}$ which satisfies the constrains above, namely the one with $\bar{\rho}_{\mathrm{id}}(c) = \rho(c_{s,i})$ and $\bar{\rho}_{\mathrm{id}}(c') = \rho(c_{s,i+1})$. It follows trivially that $\rho \models c_{s,i} = c_{s,i+1}$ iff $\bar{\rho}_{\mathrm{id}} \models c = c'$, that is, iff $\bar{\rho} \models \overline{id}(s, j, i)$.

Based on the above, and on the fact that $seq(\bar{s}, j)$ holds iff $\bar{s}$ is a number associated to a sequence s of length $2j$, we conclude that

$$\begin{aligned}
& \rho \models path_{k,m} \\
& \bar{\rho} \models \forall \bar{s} \forall j \forall i\ (seq(\bar{s}, j) \wedge j = k \wedge 0 \le i < m \rightarrow \overline{id}(\bar{s}, j, i)) \\
& \qquad \wedge \forall \bar{s} \forall j \forall i\ (seq(\bar{s}, j) \wedge j < k \wedge 0 \le i < m \\
& \qquad\qquad \rightarrow \bigvee_{\mu \in \mathcal{S}} \overline{rule}_\mu(\bar{s}, j, i) \vee \overline{id}(\bar{s}, j, i))
\end{aligned}$$

Finally, notice that by condition (2), the predicates $\beta(a, b, 0, \alpha(\rho(c_0)))$ and $\beta(a, b, m, \alpha(\rho(c_m)))$ hold. Thus, we have that

$$\begin{aligned}
\bar{\rho} \models\ & \exists c_0\ (\beta(a, b, 0, \alpha(c_0)) \wedge c = c_0) \\
& \wedge \exists c_m\ (\beta(a, b, m, \alpha(c_m)) \wedge c' = c_m)
\end{aligned}$$

iff $\rho(c_0) = \bar{\rho}(c)$ and $\rho(c_m) = \bar{\rho}(c')$, that is, iff $\rho \models c = c_0 \wedge c' = c_m$. Therefore, we conclude that statements (3) and (4) are equivalent.

To complete the proof of the lemma, we show for each ρ' that $\rho' \models path(c, c')$ iff $\rho' \models \overline{path}(c, c')$. We have that $\rho' \models path(c, c')$ iff there exist some $k, m \in \mathbb{N}$ and some $\rho : Var \rightarrow \mathcal{T}$ which agrees with ρ' on c, c' such that statement (3) holds. We also have that $\rho' \models \overline{path}(c, c')$ iff there exist some $k, m, a, b \in \mathbb{N}$ and some $\bar{\rho} : Var \rightarrow \mathcal{T}$ which agrees with ρ' on c, c' such that statement (4) holds. We know that (3) and (4) are equivalent for Gödel pairs, hence it suffices to construct for each ρ satisfying (3) a $\bar{\rho}$ such that $\rho, \bar{\rho}$ are a Gödel pair, and conversely, to construct for each $\bar{\rho}$ satisfying (4) a ρ such that $\rho, \bar{\rho}$ are a Gödel pair. For both directions, we choose such that ρ and $\bar{\rho}$ agree on c, c', k, m, thus we satisfy (1). From ρ, we construct $\bar{\rho}$ by choosing a, b to be the Gödel numbers associated to the sequence $(\alpha(c_{s,i}))$, where each indexing sequence s is of length $2j$ with $0 \le j \le k$ and for each i is such that $0 \le i \le m$ (the numbers are ordered in a sequence according to the numbers associated to the sequences s, i). Conversely, from $\bar{\rho}$, (4) implies that $\bar{\rho} \models \exists c''\ (\beta(a,\ b,\ \bar{s} + i \times p^{2\times j},\ \alpha(c'')))$ for each number $\bar{s}$ associated to an indexing sequence s of length $2j$ with $0 \le j \le k$ and for each $0 \le i \le m$. Then we choose ρ such that $\rho(c_{s,i})$ is the unique configurations in which c'' is instantiated in the formula above. In both cases, the pair $\rho, \bar{\rho}$ satisfies (2), and we are done. □

The encoding of $diverge(c)$ follows the same pattern as that of $path(c, c')$: it replaces the quantification over a sequence of configurations with the quantification over a and b, it replaces the big conjunction with the universal quantification over i and the restriction $0 \le i < m$, and uses β and α to ensure the locally quantified variables c_i and c_{i+1} are instantiated consistently. Formally, we have the following result:

Lemma 21. $\models diverge(c) \leftrightarrow \overline{diverge}(c)$.

Proof. The proof takes a similar approach to the previous one. Let $\rho, \bar{\rho} : Var \rightarrow \mathcal{T}$. We call $\rho, \bar{\rho}$ a *Gödel pair* iff

(1) ρ and $\bar{\rho}$ agree on c, m
(2) for each $0 \le i \le \rho(m)$ it is the case that

$$\beta(\bar{\rho}(a),\ \bar{\rho}(b),\ i,\ \alpha(\rho(c_i)))$$

holds. Intuitively, this states that the number associated to the configuration $\rho(c_i)$ by the injective function α is on the position i in the sequence of natural numbers Gödelized by the pair $\bar{\rho}(a), \bar{\rho}(b)$.

We intend to use ρ to interpret the top-level quantifiers in $diverge(c)$ and $\bar{\rho}$ to interpret the top-level quantifiers in $\overline{diverge}(c)$. To simplify the notation, for variables a of sort $\mathbb{N}$, like, m, a, b, i, etc, we use its syntactic name, e.g. m, to also refer to its interpretation, e.g. $\rho(k)$. Condition (1) above ensures that variables common to both $diverge(c)$ and $\overline{diverge}(c)$ are interpreted consistently by ρ and $\bar{\rho}$, and by valuations which agree with ρ or $\bar{\rho}$ on these common variables. With this convention, the instance of the β predicate above becomes

$$\beta(a,\ b,\ i,\ \alpha(\rho(c_i)))$$

We prove that for each Gödel pair $\rho, \bar{\rho}$, the following two statements are equivalent

$$\rho \models c = c_0 \wedge \bigwedge_{0 \le i < m} succ(c_i, c_{i+1}) \tag{3}$$

$$\begin{aligned}\bar{\rho} \models{}& \exists c_0\, (\beta(a,b,0,\alpha(c_0)) \wedge c = c_0) \\ &\wedge \forall i\, (0 \le i < m \to \exists c_i \exists c_{i+1}\, (\beta(a,b,i,\alpha(c_i)) \\ &\qquad \wedge \beta(a,b,i+1,\alpha(c_{i+1})) \wedge succ(c_i,c_{i+1})))\end{aligned} \tag{4}$$

First, we prove that $\rho \models succ(c_i, c_{i+1})$ iff

$$\begin{aligned}\bar{\rho} \models{}& \exists c_i \exists c_{i+1}\, (\beta(a,b,i,\alpha(c_i)) \wedge \beta(a,b,i+1,\alpha(c_{i+1})) \\ &\wedge succ(c_i,c_{i+1}))\end{aligned} \tag{5}$$

for each $0 \le i < m$. Let $\bar{\rho}' : Var \to \mathcal{T}$ which agrees with $\bar{\rho}$ on $Var \setminus \{c_i, c_{i+1}\}$. Then we have that $\bar{\rho}'$ satisfies the constrains on the configuration variables namely

$$\bar{\rho}' \models \beta(a,b,i,\alpha(c_i)) \wedge \beta(a,b,i+1,\alpha(c_{i+1}))$$

iff $\beta(a,b,i,\alpha(\bar{\rho}'(c_i)))$ and $\beta(a,b,i+1,\alpha(\bar{\rho}'(c_{i+1})))$ hold. Since ρ and $\bar{\rho}$ are a Gödel pair, condition (2) implies that $\beta(a,b,i,\alpha(\rho(c_i)))$ and $\beta(a,b,i+1,\alpha(\rho(c_{i+1})))$ hold. Further, since α is injective, and the first three arguments of β uniquely determine the fourth argument, we can conclude there exists a unique $\bar{\rho}'$ which satisfies the constrains above, namely the one with $\bar{\rho}'(c_i) = \rho(c_i)$ and $\bar{\rho}'(c_{i+1}) = \rho(c_{i+1})$. Then, it follows that $\rho \models succ(c_i, c_{i+1})$ iff $\bar{\rho}'$ satisfies

$$\bar{\rho}' \models \beta(a,b,i,\alpha(c_i)) \wedge \beta(a,b,i+1,\alpha(c_{i+1})) \wedge succ(c_i,c_{i+1})$$

that is, iff $\bar{\rho}$ satisfies (5).

Based on the above, we can conclude that

$$\rho \models \bigwedge_{0 \le i < m} succ(c_i, c_{i+1})$$

$$\begin{aligned}\bar{\rho} \models{}& \forall i\, (0 \le i < m \to \exists c_i \exists c_{i+1}\, (\beta(a,b,i,\alpha(c_i)) \\ &\wedge \beta(a,b,i+1,\alpha(c_{i+1})) \wedge succ(c_i,c_{i+1})))\end{aligned}$$

Finally, notice that by condition (2), the predicates $\beta(a,b,0,\alpha(\rho(c_0)))$ holds. Thus, we have that

$$\bar{\rho} \models \exists c_0\, (\beta(a,b,0,\alpha(c_0)) \wedge c = c_0)$$

iff $\rho(c_0) = \bar{\rho}(c)$, that is, iff $\rho \models c = c_0$. Therefore, we conclude that statements (3) and (4) are equivalent.

To complete the proof of the lemma, we show for each ρ' that $\rho' \models diverge(c)$ iff $\rho' \models \overline{diverge}(c)$. We have that $\rho' \models diverge(c)$ iff for each $m \in \mathbb{N}$ there exists some $\rho : Var \to \mathcal{T}$ which agrees with ρ' on c such that statement (3) holds. We also have that $\rho' \models \overline{diverge}(c)$ iff for each $m \in \mathbb{N}$ there exist some $a, b \in \mathbb{N}$ and some $\bar{\rho} : Var \to \mathcal{T}$ which agrees with ρ' on c such that statement (4) holds. We know that (3) and (4) are equivalent for Gödel pairs, hence it suffices to construct for each ρ satisfying (3) a $\bar{\rho}$ such that $\rho, \bar{\rho}$ are a Gödel pair and agree on m, and conversely, to construct for each $\bar{\rho}$ satisfying (4) a ρ such that $\rho, \bar{\rho}$ are a Gödel pair and agree m. For both directions, we choose such that ρ and $\bar{\rho}$ agree on c, m, thus we satisfy (1). From ρ, we construct $\bar{\rho}$ by choosing a, b to be the Gödel numbers associated to the sequence $\alpha(c_0), \ldots, \alpha(c_m)$. Conversely, from $\bar{\rho}$, (4) implies that $\bar{\rho} \models \exists c'\, (\beta(a,\ b,\ i,\ \alpha(c')))$ for each number $0 \le i \le m$. Then we choose ρ such that $\rho(c_i)$ is the unique configurations in which c'' is instantiated in the formula above. In both cases, the pair $\rho, \bar{\rho}$ satisfies (2), and we are done. □

The following summarises the two results above:

Lemma 22. *$\models path(c,c') \leftrightarrow \overline{path}(c,c')$ and $\models diverge(c) \leftrightarrow \overline{diverge}(c)$.*

Proof. Follows by Lemma 20 and Lemma 21. □

Consequently, we can use $\overline{path}(c,c')$, to express $step(c,c')$, $succ(c,c')$ and $coreach(\varphi)$ in FOL. Then, $\overline{diverge}(c)$, which uses $succ(c,c')$, is a FOL formula. Since our relative completeness proof only uses these predicates besides other FOL formulae over the signature Σ, we can conclude that all the formulae used in our proof are FOL formulae. For notational simplicity, we continue working with $path$ and $diverge$ instead of $\overline{path}$ and $\overline{diverge}$.

6.4 Semantic Validity and Relative Completeness

We derive a proof for a valid rule $\varphi \Rightarrow \varphi'$ with the proof system in Figure 3, using the FOL predicates encoding the transition system to express intermediate formulae. First, we show that the semantic validity of reachability rules can be framed as FOL validity. This does not give us relative completeness directly, but it enables us to begin the derivation process.

Lemma 23. *$\mathcal{S} \models \varphi \Rightarrow \varphi'$ iff $\models \varphi \to \exists c\, (\square = c \wedge diverge(c)) \vee coreach(\varphi')$.*

Proof. Let $\gamma \in \mathcal{T}_{Cfg}$ and $\rho : Var \to \mathcal{T}$. We establish the necessary and sufficient conditions such that

$$(\gamma, \rho) \models \varphi \to \exists c\, (\square = c \wedge diverge(c)) \vee coreach(\varphi')$$

According to the semantics of implications, the above happens iff $(\gamma, \rho) \models \varphi$ implies either $(\gamma, \rho) \models \exists c\, (\square = c \wedge diverge(c))$ or $(\gamma, \rho) \models coreach(\varphi')$. By Lemma 16, the former holds iff γ does not terminate, while by Lemma 19 the latter holds iff there exists some $\gamma' \in \mathcal{T}_{Cfg}$ such that $(\gamma', \rho) \models \varphi'$ and $\gamma \to_{\mathcal{S}}^{\star} \gamma'$. Thus, we can conclude that $\models \varphi \to \exists c\, (\square = c \wedge diverge(c)) \vee coreach(\varphi')$ iff for each γ and ρ we have that $(\gamma, \rho) \models \varphi$ implies that either γ does not terminate or there exists some $\gamma' \in \mathcal{T}_{Cfg}$ such that $(\gamma', \rho) \models \varphi'$ and $\gamma \to_{\mathcal{S}}^{\star} \gamma'$. According to the definition of semantic validity for reachability rules (see Definition 10), the above conditions hold iff $\mathcal{S} \models \varphi \Rightarrow \varphi'$, and we are done. □

The following three lemmas are useful in proving our relative completeness result.

Lemma 24. *If $\mathcal{A} \vdash \square = c \wedge path(c,c') \Rightarrow \square = c'$ is derivable and φ is well-defined, then $\mathcal{A} \vdash \varphi \wedge \varphi[c/\square] \wedge \varphi'[c'/\square] \wedge path(c,c') \Rightarrow \varphi'$ is derivable.*

Proof. Since φ is well-defined, we have that $\models \varphi \wedge \varphi[c/\square] \to \square = c$, by Lemma 8. Also, it is easy to see that $\models \square = c' \wedge \varphi'[c'/\square] \to \varphi'$. Thus, $\mathcal{A} \vdash \varphi \wedge \varphi[c/\square] \wedge \varphi'[c'/\square] \wedge path(c,c') \Rightarrow \varphi'$ is derivable by Consequence with the two implications above from the sequent

$$\mathcal{A} \vdash \square = c \wedge \varphi'[c'/\square] \wedge path(c,c') \Rightarrow \square = c' \wedge \varphi'[c'/\square]$$

which follows by Logical Framing (Lemma 3) with $\varphi'[c'/\square]$ from

$$\mathcal{A} \vdash \square = c \wedge path(c,c') \Rightarrow \square = c'$$

The last sequent is derivable according to the hypothesis, and we are done. □

Lemma 25. *If $\mathcal{A} \vdash \square = c \wedge step(c,c'') \wedge path(c'',c') \Rightarrow \square = c'$ is derivable, then $\mathcal{A} \vdash \square = c \wedge path(c,c') \Rightarrow \square = c'$ is derivable.*

Proof. By Lemma 14, we have that

$$\models path(c, c') \to c = c' \vee \exists c''\ (step(c, c'') \wedge path(c'', c'))$$

Then, $\mathcal{A} \vdash \square = c \wedge path(c, c') \Rightarrow \square = c'$ is derivable by Consequence with the implication above, Abstraction with c'', and Case Analysis from

$$\mathcal{A} \vdash \square = c \wedge c = c' \Rightarrow \square = c'$$
$$\mathcal{A} \vdash \square = c \wedge step(c, c'') \wedge path(c'', c') \Rightarrow \square = c'$$

The former follows by Consequence and Reflexivity, while the latter is derivable according to the hypothesis, and we are done. □

Lemma 26. *If $\mathcal{S} \cup C \vdash \square = c \wedge path(c, c') \Rightarrow \square = c'$ is derivable, then $\mathcal{S} \vdash_C \square = c \wedge step(c, c') \Rightarrow \square = c'$ is derivable.*

Proof. The sequent $\mathcal{S} \vdash_C \square = c \wedge step(c, c') \Rightarrow \square = c'$ follows by Abstraction with $c_1, c'_1, \ldots, c_{nc}, c'_{nc}, \bar{x}$ (the top existentially quantified variables in the definition of $step(c, c')$) from

$$\mathcal{S} \vdash_C \bigvee_{\mu \in \mathcal{S}} (\varphi[c/\square] \wedge \varphi'[c'/\square] \wedge \bigwedge_{1 \le i \le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square] \wedge path(c_i, c'_i)))$$
$$\wedge \square = c$$
$$\Rightarrow \square = c$$

Then, by Consequence, we can drop $\square = c$ and substitute c by $\square$, and it suffices to derive

$$\mathcal{S} \vdash_C \bigvee_{\mu \in \mathcal{S}} (\varphi \wedge \varphi'[c'/\square] \wedge \bigwedge_{1 \le i \le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square] \wedge path(c_i, c'_i)))$$
$$\Rightarrow \square = c$$

Recall that one of the assumptions of relative completeness is that $\mathcal{S}$ is not empty. Then, by $|\mathcal{S}| - 1$ applications of Case Analysis, it suffices to derive for each $\mu \in \mathcal{S}$

$$\mathcal{S} \vdash_C \varphi \wedge \varphi'[c'/\square] \wedge \bigwedge_{1 \le i \le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square] \wedge path(c_i, c'_i)) \Rightarrow \square = c$$

Recall that one of the assumptions of relative completeness is that φ' is well-defined. By Lemma 8, $\models \varphi' \wedge \varphi'[c'/\square] \to \square = c'$, thus by Consequence it suffices to derive

$$\mathcal{S} \vdash_C \varphi \wedge \varphi'[c'/\square] \wedge \bigwedge_{1 \le i \le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square] \wedge path(c_i, c'_i))$$
$$\Rightarrow \varphi' \wedge \varphi'[c'/\square]$$

The last sequent follows by Logical Framing (Lemma 3) with $\varphi'[c'/\square]$ and Axiom with $\mu \in \mathcal{S}$ from the prerequisites

$$\mathcal{S} \cup C \vdash \varphi_j \wedge \bigwedge_{1 \le i \le n} (\varphi_i[c_i/\square] \wedge \varphi'_i[c'_i/\square] \wedge path(c_i, c'_i)) \Rightarrow \varphi'_j$$

for each $1 \le j \le n$. By Consequence, it suffices to derive

$$\mathcal{S} \cup C \vdash \varphi_j \wedge \varphi_j[c_j/\square] \wedge \varphi'_j[c'_j/\square] \wedge path(c_j, c'_j) \Rightarrow \varphi'_j$$

According to the hypothesis, $\mathcal{S} \cup C \vdash \square = c \wedge path(c, c') \Rightarrow \square = c'$ is derivable. Recall that one of the assumptions of relative completeness is that φ_j is well-defined. Then the last sequent follows by Lemma 24 with α-renaming c, c' into c_j, c'_j, and we are done. □

The following lemma states that if there is a path in the transition system from c to c' (expressed by $path(c, c')$) then we can derive it.

Lemma 27. $\mathcal{S} \vdash \square = c \wedge path(c, c') \Rightarrow \square = c'$.

Proof. By Lemma 25, it suffices to derive

$$\mathcal{S} \vdash \square = c \wedge step(c, c'') \wedge path(c'', c') \Rightarrow \square = c'$$

Let $C \equiv \{\square = c \wedge step(c, c'') \wedge path(c'', c') \Rightarrow \square = c'\}$. Then, by Circularity, it suffices to derive

$$\mathcal{S} \vdash_C \square = c \wedge step(c, c'') \wedge path(c'', c') \Rightarrow \square = c'$$

which in turn follows by Transitivity from

$$\mathcal{S} \vdash_C \square = c \wedge step(c, c'') \wedge path(c'', c') \Rightarrow \square = c'' \wedge path(c'', c') \quad (1)$$
$$\mathcal{S} \cup C \vdash \square = c'' \wedge path(c'', c') \Rightarrow \square = c' \quad (2)$$

Sequent (1) follows by Logic Framing with $path(c'', c')$ from

$$\mathcal{S} \vdash_C \square = c \wedge step(c, c'') \Rightarrow \square = c''$$

By Lemma 26 (with α-renaming c'' into c'), it suffices to derive

$$\mathcal{S} \cup C \vdash \square = c \wedge path(c, c') \Rightarrow \square = c'$$

Further, by Lemma 25, it suffices to derive

$$\mathcal{S} \cup C \vdash \square = c \wedge step(c, c'') \wedge path(c'', c') \Rightarrow \square = c'$$

which follows by Axiom with the rule in C. By Lemma 25 (with α-renaming c'' into c), sequent (2) follows from

$$\mathcal{S} \cup C \vdash \square = c \wedge step(c, c'') \wedge path(c'', c') \Rightarrow \square = c'$$

which follows by Axiom with the rule in C, and we are done. □

The result above has the following consequence:

Lemma 28. $\mathcal{S} \vdash_C \square = c \wedge step(c, c') \Rightarrow \square = c'$.

Proof. By Lemma 26, it suffices to derive

$$\mathcal{S} \cup C \vdash \square = c \wedge path(c, c') \Rightarrow \square = c'$$

which follows form Lemma 27 and weakening (Lemma 2) with $\mathcal{S} \subseteq \mathcal{S} \cup C$. □

The next lemma states that if c diverges in the transition system then we can derive it.

Lemma 29. $\mathcal{S} \vdash \square = c \wedge diverge(c) \Rightarrow \omega$.

Proof. Let $C \equiv \{\square = c \wedge diverge(c) \Rightarrow \omega\}$. Then the sequent $\mathcal{S} \vdash \square = c \wedge diverge(c) \Rightarrow \omega$ follows by Circularity from

$$\mathcal{S} \vdash_C \square = c \wedge diverge(c) \Rightarrow \omega$$

By Lemma 17, we have that

$$\models diverge(c) \to \exists c'\ (succ(c, c') \wedge diverge(c'))$$

Thus, by Consequence, and Abstraction with c', it suffices to derive

$$\mathcal{S} \vdash_C \square = c \wedge succ(c, c') \wedge diverge(c') \Rightarrow \omega$$

Let ψ_{cond} be defined as

$$\psi_{cond} \equiv \bigvee_{\mu \in \mathcal{S}} (\varphi[c/\square] \wedge \bigvee_{1 \le i \le n} (\varphi_i[c'/\square]$$
$$\wedge \bigwedge_{1 \le j < i} (\varphi_j[c_j/\square] \wedge \varphi'_j[c'_j/\square] \wedge path(c_j, c'_j))))$$

Then, according to the definition of $succ(c, c')$, by Case Analysis and Abstraction with $c_1, c'_1, \ldots, c_{nc}, c'_{nc}, \bar{x}$ (the top existentially quantified variables), it suffice to derive

$$\mathcal{S} \vdash_C \square = c \wedge step(c, c') \wedge diverge(c') \Rightarrow \omega \quad (1)$$
$$\mathcal{S} \vdash_C \square = c \wedge \psi_{cond} \wedge diverge(c') \Rightarrow \omega \quad (2)$$

By Lemma 28 and Logic Framing with $diverge(c')$, we derive

$$\mathcal{S} \vdash_C \square = c \wedge step(c, c') \wedge diverge(c') \Rightarrow \square = c' \wedge diverge(c')$$

By Axiom with the rule in C and α-renaming of c' into c, we derive

$$\mathcal{S} \cup C \vdash \square = c' \wedge diverge(c') \Rightarrow \omega$$

Then, sequent (1) follows by Transitivity with the two sequents above. Therefore, we are left to derive sequent (2). Let φ_{cond} be defined as

$$\varphi_{cond} \equiv \bigvee_{\mu \in \mathcal{S}} (\varphi \wedge \bigvee_{1 \leq i \leq n} (\varphi_i[c'/\square] \\ \wedge \bigwedge_{1 \leq j < i} (\varphi_j[c_j/\square] \wedge \varphi'_j[c'_j/\square] \wedge path(c_j, c'_j))))$$

Then, by Consequence, we can drop $\square = c$ and substitute c by $\square$ in sequent (2), and it suffices to derive

$$\mathcal{S} \vdash_C \varphi_{cond} \wedge diverge(c') \Rightarrow \omega$$

Recall that one of the assumptions of relative completeness is that $\mathcal{S}$ is not empty. Then, by $|\mathcal{S}| - 1$ applications of Case Analysis, each followed by n applications of Case Analysis, it suffices to derive for each $\mu \in \mathcal{S}$ and each $1 \leq i \leq n$ (n is the number of conditions of μ)

$$\mathcal{S} \vdash_C \varphi \wedge \varphi_i[c'/\square] \wedge \bigwedge_{1 \leq j < i} (\varphi_j[c_j/\square] \wedge \varphi'_j[c'_j/\square] \wedge path(c_j, c'_j)) \\ \wedge diverge(c') \\ \Rightarrow \omega$$

Recall that one of the assumption of relative completeness is that $\mathcal{S}$ is ω-closed (see Definition 11). Thus, for μ and i there must be some rule $\mu_\omega \in \mathcal{S}$ of the form

$$\varphi \Rightarrow \omega \text{ if } \varphi_1 \Rightarrow \varphi'_1 \bigwedge \ldots \bigwedge \varphi_{i-1} \Rightarrow \varphi'_{i-1} \bigwedge \varphi_i \Rightarrow \omega$$

Then the sequent above follows by Axiom with μ_ω and with the prerequisites

$$\mathcal{S} \cup C \vdash \varphi_k \wedge \varphi_i[c'/\square] \wedge \bigwedge_{1 \leq j < i} (\varphi_j[c_j/\square] \wedge \varphi'_j[c'_j/\square] \wedge path(c_j, c'_j)) \\ \wedge diverge(c') \\ \Rightarrow \varphi'_k \tag{3}$$

$$\mathcal{S} \cup C \vdash \varphi_i \wedge \varphi_i[c'/\square] \wedge \bigwedge_{1 \leq j < i} (\varphi_j[c_j/\square] \wedge \varphi'_j[c'_j/\square] \wedge path(c_j, c'_j)) \\ \wedge diverge(c') \\ \Rightarrow \omega \tag{4}$$

for each $1 \leq k < i$. By Consequence, to derive each sequent (3), it suffices to derive

$$\mathcal{S} \cup C \vdash \varphi_k \wedge \varphi_k[c_k/\square] \wedge \varphi'_k[c'_k/\square] \wedge path(c_k, c'_k) \Rightarrow \varphi'_k$$

for each $1 \leq k < i$. By Lemma 27, we have that

$$\mathcal{S} \vdash \square = c \wedge path(c, c') \Rightarrow \square = c'$$

is derivable. Recall that one of the assumption of relative completeness is that φ_k is well-defined. Then, the sequents above follow by Lemma 24 with α-renaming c, c into c_k, c'_k. By Consequence, to derive sequent (4), it suffices to derive

$$\mathcal{S} \cup C \vdash \varphi_i \wedge \varphi_i[c'/\square] \wedge diverge(c') \Rightarrow \omega$$

Recall that one of the assumption of relative completeness is that φ_i is well-defined. By Lemma 8, we have that $\models \varphi_i \wedge \varphi_i[c'/\square] \rightarrow \square = c'$. Thus, by Consequence the last sequent follows from

$$\mathcal{S} \cup C \vdash \square = c' \wedge diverge(c') \Rightarrow \omega$$

which follows by Axiom with the rule in C and α-renaming c into c', and we are done. □

Lemma 30. *$\mathcal{S} \vdash coreach(\varphi) \Rightarrow \varphi$.*

Proof. According to the definition of *coreach*(φ), by Abstraction with c, c', it suffices to derive

$$\mathcal{S} \vdash \square = c \wedge \varphi[c'/\square] \wedge path(c, c') \Rightarrow \varphi$$

We have that $\models \square = c' \wedge \varphi[c'/\square] \rightarrow \varphi'$, thus the last sequent follows by Consequence from

$$\mathcal{S} \vdash \square = c \wedge \varphi[c'/\square] \wedge path(c, c') \Rightarrow \square = c' \wedge \varphi[c'/\square]$$

which in turn follows by Logic Framing (see Lemma 3) with $\varphi[c'/\square]$ from

$$\mathcal{S} \vdash \square = c \wedge path(c, c') \Rightarrow \square = c'$$

which is derivable by Lemma 27, and we are done. □

Using the lemmas above, we derive the following rule between the formula specifying the configurations reaching φ and the divergent configurations, on one hand, and φ itself, on the other hand.

Lemma 31. *$\mathcal{S} \vdash \exists c\ (\square = c \wedge diverge(c)) \vee coreach(\varphi) \Rightarrow \varphi$.*

Proof. By Case Analysis and Abstraction with c, it suffices to derive

$$\mathcal{S} \vdash \square = c \wedge diverge(c)) \Rightarrow \varphi$$
$$\mathcal{S} \vdash coreach(\varphi) \Rightarrow \varphi$$

The latter follows by Lemma 30. For former follows by Transitivity from

$$\mathcal{S} \vdash \square = c \wedge diverge(c)) \Rightarrow \omega$$
$$\mathcal{S} \vdash \omega \Rightarrow \varphi$$

The first sequent is derivable by Lemma 29. To derive the second, Circularity and Transitivity, it suffices to derive

$$\mathcal{S} \vdash_{\{\omega \Rightarrow \varphi\}} \omega \Rightarrow \omega$$
$$\mathcal{S} \cup \{\omega \Rightarrow \varphi\} \vdash \omega \Rightarrow \varphi$$

Recall that one of the assumption of relative completeness is that $\mathcal{S}$ is ω-closed. Then the first sequent follows by Axiom with $\omega \Rightarrow \omega$, while the second sequent follows by Axiom with $\omega \Rightarrow \varphi$, and we are done. □

Finally, the relative completeness follows from all the lemmas above. Note how the configuration model is being used, via Lemma 23, as an oracle to answer the semantic reachability question formulated as a FOL sentence.

Theorem 2. *If $\mathcal{S} \models \varphi \Rightarrow \varphi'$ then $\mathcal{S} \vdash \varphi \Rightarrow \varphi'$.*

Proof. Suppose that $\mathcal{S} \models \varphi \Rightarrow \varphi'$. Then Lemma 23 implies that $\models \varphi \rightarrow (\exists c\ (\square = c \wedge diverge(c)) \vee coreach(\varphi'))$. By Lemma 31 it follows that $\mathcal{S} \vdash \exists c\ (\square = c \wedge diverge(c)) \vee coreach(\varphi') \Rightarrow \varphi'$ is derivable. Then the theorem follows by Consequence. □

A direct consequence of the theorem above is the following (recall that for relative completeness $\mathcal{S}$ is assumed ω-closed)

Corollary 2. *If $\mathcal{S} \models \varphi\uparrow$ then $\mathcal{S} \vdash \varphi \Rightarrow \omega$.*

7. Conclusion and Future Work

We presented *reachability logic*, a novel framework for reasoning about reachability which unifies operational and axiomatic semantics. The reachability logic sentences are called *reachability rules*, and can express both transitions between configurations, as needed for operational semantics, and Hoare-style triples, as needed for axiomatic semantics. A programming language is given as a set of reachability rules defining its operational semantics, and a language-independent seven-rule proof system is then used to derive any reachability property of the language. Our reachability logic proof system was proven sound and relatively complete. The soundness was also mechanized in Coq, which lets reachability logic proofs serve as proof certificates.

Until this paper, reachability rules were unconditional, so they could only express a limited number of operational semantics (whose rules had no premises). By allowing *conditional* rules, reachability logic can now express virtually all operational semantics styles, including the standard small-step (by Plotkin) and big-step (by Kahn) semantics which were excluded before.

We hope reachability logic may serve a foundation for verifying programs. Also, since Hoare-style proof derivations can be mechanically translated into reachability logic proof derivations, reachability logic may serve as a more elegant means to establish the soundness of axiomatic semantics. We plan to extend our MatchC prover and connect it to the Coq formalization to produce a mostly-automated and efficient certifying verifier based on the presented proof system.

References

[1] A. W. Appel. Verified software toolchain. In *ESOP*, volume 6602 of *LNCS*, pages 1–17, 2011.

[2] G. Berry and G. Boudol. The chemical abstract machine. *Th. Comp. Sci.*, 96(1):217–248, 1992.

[3] S. Blazy and X. Leroy. Mechanized semantics for the Clight subset of the C language. *J. Autom. Reasoning*, 43(3):263–288, 2009.

[4] M. Clavel, F. Durán, S. Eker, J. Meseguer, P. Lincoln, N. Martí-Oliet, and C. Talcott. *All About Maude*, volume 4350 of *LNCS*. 2007.

[5] D. Clément, J. Despeyroux, L. Hascoet, and G. Kahn. Natural semantics on the computer. In K. Fuchi and M. Nivat, editors, *Proceedings of the France-Japan AI and CS Symposium*, pages 49–89. ICOT, Japan, 1986.

[6] T. Coquand and G. Huet. The calculus of constructions. *Inf. Comput.*, 76(2-3):95–120, 1988.

[7] T.-F. Şerbanuţă, G. Roşu, and J. Meseguer. A rewriting logic approach to operational semantics. *Inf. & Computation*, 207(2):305–340, 2009.

[8] C. Ellison and G. Roşu. An executable formal semantics of C with applications. In *POPL*, pages 533–544, 2012.

[9] A. Farzan, F. Chen, J. Meseguer, and G. Roşu. Formal analysis of Java programs in JavaFAN. In *CAV'04*, volume 3114 of *LNCS*, pages 501–505, 2004.

[10] M. Felleisen and D. P. Friedman. Control operators, the SECD-machine, and the lambda-calculus. In *3rd Working Conference on the Formal Description of Programming Concepts*, pages 193–219, Ebberup, Denmark, Aug. 1986.

[11] R. W. Floyd. Assigning meaning to programs. In *Symposium on Applied Mathematics*, volume 19, pages 19–32, 1967.

[12] C. George, A. E. Haxthausen, S. Hughes, R. Milne, S. Prehn, and J. S. Pedersen. *The RAISE Development Method*. Prentice Hall, 1995.

[13] J. Giesl and T. Arts. Verification of erlang processes by dependency pairs. *Appl. Algebra Eng. Commun. Comput.*, 12(1/2):39–72, 2001.

[14] J. Goguen and G. Malcolm. *Algebraic Semantics of Imperative Programs*. MIT Press, 1996.

[15] D. Harel, D. Kozen, and J. Tiuryn. Dynamic logic. In *Handbook of Philosophical Logic*, pages 497–604, 1984.

[16] C. A. R. Hoare and H. Jifeng. *Unifying Theories of Programming*. Prentice Hall, 1998.

[17] B. Jacobs. Weakest pre-condition reasoning for java programs with JML annotations. *J. Log. Algebr. Program.*, 58(1-2):61–88, 2004.

[18] D. Leivant. Higher order logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming (2)*, pages 229–322. Oxford University Press, 1994.

[19] H. Liu and J. S. Moore. Java program verification via a JVM deep embedding in ACL2. In *TPHOLs*, volume 3223 of *LNCS*, pages 184–200, 2004.

[20] S. Lucas, C. Marché, and J. Meseguer. Operational termination of conditional term rewriting systems. *Inf. P. Ltrs.*, 95(4):446–453, 2005.

[21] The Coq development team. *The Coq proof assistant reference manual*. LogiCal Project, 2004. URL http://coq.inria.fr. Version 8.0.

[22] P. O. Meredith, M. Katelman, J. Meseguer, and G. Roşu. A formal executable semantics of Verilog. In *MEMOCODE'10*, pages 179–188. IEEE, 2010.

[23] J. Meseguer. Conditioned rewriting logic as a united model of concurrency. *Theor. Comput. Sci.*, 96(1):73–155, 1992.

[24] J. Meseguer and C. Braga. Modular rewriting semantics of programming languages. In *AMAST*, volume 3116 of *LNCS*, pages 364–378, 2004.

[25] P. D. Mosses. *CASL Reference Manual*, volume 2960 of *LNCS*. Springer, 2004.

[26] T. Nipkow. Winskel is (almost) right: Towards a mechanized semantics textbook. *Formal Aspects of Computing*, 10:171–186, 1998.

[27] P. W. O'Hearn and D. J. Pym. The logic of bunched implications. *Bulletin of Symb. Logic*, 5(2):215–244, 1999.

[28] D. Pavlovic and D. R. Smith. Composition and refinement of behavioral specifications. In *ASE*, pages 157–165, 2001.

[29] G. Plotkin. A structural approach to operational semantics. *Journal of Logic & Algebraic Programming*, 60-61:17–139, 2004.

[30] J. C. Reynolds. Separation logic: A logic for shared mutable data structures. In *LICS*, pages 55–74, 2002.

[31] G. Roşu and A. Ştefănescu. Matching logic: a new program verification approach (NIER track). In *ICSE*, pages 868–871, 2011.

[32] G. Roşu and A. Ştefănescu. From hoare logic to matching logic reachability. In *FM'12*, LNCS. Springer, 2012. To appear.

[33] G. Roşu and A. Ştefănescu. Towards a unified theory of operational and axiomatic semantics. In *ICALP'12*, LNCS. Springer, 2012. To appear.

[34] G. Roşu and A. Ştefănescu. Checking reachability using matching logic. In *OOPSLA*. ACM, 2012. To appear.

[35] G. Roşu, C. Ellison, and W. Schulte. Matching logic: An alternative to Hoare/Floyd logic. In *AMAST*, volume 6486 of *LNCS*, pages 142–162, 2010.

[36] R. Sasse and J. Meseguer. Java+ITP: A verification tool based on Hoare logic and algebraic semantics. *ENTCS*, 176(4):29–46, 2007.

[37] G. Winskel. *The formal semantics of programming languages - an introduction*. Foundation of computing series. MIT Press, 1993.

[38] A. Wright and M. Felleisen. A syntactic approach to type soundness. *Inf. & Computation*, 115(1):38–94, 1994.