Abstract Framework for All-Path Reachability Analysis toward Safety and Liveness Verification

Misaki Kojima
Graduate School of Informatics, Nagoya University, Japan

Naoki Nishida
Graduate School of Informatics, Nagoya University, Japan

Abstract

An all-path reachability (APR, for short) predicate over an object set is a pair of a source set and a target set, which are subsets of the object set. APR predicates have been defined for abstract reduction systems (ARSs, for short) and then extended to logically constrained term rewrite systems (LCTRSs, for short) as pairs of constrained terms that represent sets of terms modeling configurations, states, etc. An APR predicate is partially (or demonically) valid w.r.t. a rewrite system if every finite maximal reduction sequence of the system starting from any element in the source set includes an element in the target set. Partial validity of APR predicates w.r.t. ARSs is defined by means of two inference rules, which can be considered a proof system to construct (possibly infinite) derivation trees for partial validity. On the other hand, a proof system for LCTRSs consists of four inference rules, leaving a gap between the inference rules for ARSs and LCTRSs. In this paper, we revisit the framework for APR analysis and adapt it to verification of not only safety but also liveness properties. To this end, we first reformulate an abstract framework for partial validity w.r.t. ARSs so that there is a one-to-one correspondence between the inference rules for partial validity w.r.t. ARSs and LCTRSs. Secondly, we show how to apply APR analysis to safety verification. Thirdly, to apply APR analysis to liveness verification, we introduce a novel stronger validity of APR predicates, called total validity, which requires not only finite but also infinite execution paths to reach target sets. Finally, for a partially valid APR predicate with a cyclic-proof tree, we show that the acyclicity of the proof graph obtained from the cyclic-proof tree is a necessary and sufficient condition for total validity. The condition implies that if there exists a cyclic-proof tree for an APR predicate, the proof graph of which is acyclic, then the APR predicate is totally valid.

2012 ACM Subject Classification Theory of computation → Rewrite systems

Keywords and phrases abstract reduction system, reachability, cyclic proof, runtime-error verification

Digital Object Identifier 10.4230/LIPIcs.FSCD.2026.19

Related Version *Full Version*: `https://arxiv.org/abs/2602.04641` [18]

Funding This research was supported by JSPS KAKENHI Grant Number JP24K02900.
Misaki Kojima: Grant-in-Aid for JSPS Fellows Grant Number JP24KJ1240

Acknowledgements We thank the anonymous reviewers for their valuable feedback, which improved the paper.

1 Introduction

Recently, program verification approaches using *logically constrained term rewrite systems* (LCTRSs, for short) [20] have been extensively investigated [10, 26, 7, 23, 11, 12, 6, 21, 19, 22]. LCTRSs are effective models of both functional and imperative programs. For instance, equivalence checking by means of LCTRSs is useful to ensure the correctness of terminating functions (cf. [10]). Since the reduction of rewrite systems is in general non-deterministic, rewrite systems are reasonable models of concurrent programs. A transformation of sequential

$$\textsc{Subsumption}\ \frac{}{P \Rightarrow Q}\ \text{if } P \subseteq Q. \qquad \textsc{Step}\ \frac{\partial_{\mathcal{A}}(P \setminus Q) \Rightarrow Q}{P \Rightarrow Q}\ \text{if } (P \setminus Q) \text{ is } \mathcal{A}\text{-runnable.}$$

Figure 1 Inference rules of DVP [7] for partial validity of APR predicates w.r.t. an ARS $\mathcal{A} = (A, \to_{\mathcal{A}})$, where $\partial_{\mathcal{A}}(P) = \{t' \in A \mid \exists t \in P.\ t \to_{\mathcal{A}} t'\}$.

programs into LCTRSs [10, 11] has been extended to concurrent programs [19]. In addition, a method for runtime-error verification by means of *all-path reachability analysis* of LCTRSs has been developed for, e.g., race and starvation freedom [15, 14, 16].

An *all-path reachability* (APR, for short) *predicate* over an object set A is a pair $P \Rightarrow Q$ of a *source* set P and a *target* set Q, which are subsets of A. The APR predicate is said to be *partially valid* (or *demonically valid* [7, Definition 5]) w.r.t. a rewrite system $\mathcal{R}$, the reduction of which is defined over A, if every finite *execution path* – a maximal reduction sequence – of $\mathcal{R}$ starting from any element s in P includes an element in Q. Partial validity of $P \Rightarrow Q$ w.r.t. $\mathcal{R}$ means that every terminating execution from P eventually reaches Q. The *APR problem* w.r.t. $\mathcal{R}$ is a problem to determine whether given APR predicates are partially valid w.r.t. $\mathcal{R}$ or not.

An *abstract* APR framework, which is a framework for *abstract reduction systems* (ARSs, for short), was first proposed [7]. In the framework, partial validity of APR predicates w.r.t. an ARS $\mathcal{A} = (A, \to_{\mathcal{A}})$ is defined by the two rules of the inference system DVP (Demonically Valid Predicate) [7], which are shown in Figure 1. Here, the *derivative* $\partial_{\mathcal{A}}(P)$ is the set of successors of elements in P w.r.t. $\to_{\mathcal{A}}$ and a set P ($\subseteq A$) is said to be $\mathcal{A}$*-runnable* if $P \neq \emptyset$ and P does not include any normal form of $\mathcal{A}$. To be more precise, the set of partially valid APR predicates w.r.t. $\mathcal{A}$ is defined by the greatest fixed point of the functional of DVP parameterized by $\mathcal{A}$. Rule Subsumption defines trivially partially valid APR predicates $P \Rightarrow Q$ with $P \subseteq Q$, independent from $\mathcal{A}$; since $P \subseteq Q$, every execution path starting from P reaches Q because the head element of the path is in Q. Rule Step first removes the execution paths starting with elements in $P \cap Q$, which are partially valid w.r.t. $\mathcal{A}$, and then generates a subgoal $\partial_{\mathcal{A}}(P \setminus Q) \Rightarrow Q$ for the tail execution paths of those starting with elements $P \setminus Q$. Since the inference system DVP w.r.t. $\mathcal{A}$ can be used to prove partial validity of APR predicates w.r.t. $\mathcal{A}$, it can be considered a proof system to construct (possibly infinite) derivation trees of given APR predicates.

The abstract APR framework has been adapted to LCTRSs, and the proof system DCC (Demonic Circular Coinduction) for partial validity has been presented [7]. An APR predicate over a signature Σ is a pair $\langle s \mid \phi \rangle \Rightarrow \langle t \mid \psi \rangle$ of constrained terms $\langle s \mid \phi \rangle, \langle t \mid \psi \rangle$ over Σ, which represents sets of (ground) terms standing for configurations, states, etc (cf. [10, 11, 14]). To simplify the discussion, we assume that the two constrained terms have no shared variables. Note that $\mathcal{R}$ induces the ARS $(T(\Sigma), \to_{\mathcal{R}})$ for the set $T(\Sigma)$ of ground terms over Σ, and a constrained term $\langle s \mid \phi \rangle$ can be considered the set of ground normalized instances of s by means of substitutions satisfying ϕ. The proof system DCC w.r.t. an LCTRS $\mathcal{R}$ with an underlying theory $\mathcal{T}$ consists of the four inference rules shown in Figure 2, where $\Delta_{\mathcal{R}}(\langle s \mid \phi \rangle)$ is the set of successor constrained terms of $\langle s \mid \phi \rangle$ w.r.t. constrained narrowing (cf. [17, Lemma 3.5]): $\Delta_{\mathcal{R}}(\langle s \mid \phi \rangle) = \{\langle s[r]_p \mid \phi \wedge (s = s[\ell]_p) \wedge \varphi \rangle \mid (\ell \to r\ [\varphi]) \in \mathcal{R}, s|_p \text{ is not a variable}\}$ [7, Definition 11], where $\ell \to r\ [\varphi]$ is renamed so that $\mathcal{V}ar(s, \phi) \cap \mathcal{V}ar(\ell, r, \varphi) = \emptyset$. Here, G given in advance is a set of APR predicates to be proved partially valid, e.g., a main goal and other APR predicates used as auxiliary lemmas to prove the main goal. In applying Circ to $\langle s \mid \phi \rangle \Rightarrow \langle t \mid \psi \rangle$, the APR predicate $\langle s' \mid \phi' \rangle \Rightarrow \langle t' \mid \psi' \rangle$ in the side condition may be the same as $\langle s \mid \phi \rangle \Rightarrow \langle t \mid \psi \rangle$. For the soundness of DCC, Der has to be applied to all APR predicates in G. Note that the application of Subs is followed by Circ or either Axiom or Der.

$$\textsc{Axiom}\ \frac{}{\langle s \,|\, \phi\rangle \Rightarrow \langle t \,|\, \psi\rangle}\ \text{if } \phi \text{ is unsatisfiable w.r.t. } \mathcal{T}.$$

$$\textsc{Subs}\ \frac{\langle s \,|\, \phi \wedge \neg(\exists \vec{x}.\ ((s = t) \wedge \psi))\rangle \Rightarrow \langle t \,|\, \psi\rangle}{\langle s \,|\, \phi\rangle \Rightarrow \langle t \,|\, \psi\rangle}\ \text{if } (s = t) \wedge \phi \wedge \psi \text{ is satisfiable w.r.t. } \mathcal{T},$$

where $\{\vec{x}\} = \mathcal{V}ar(t, \psi) \setminus \mathcal{V}ar(s, \phi)$.

$$\textsc{Der}\ \frac{\langle s_1 \,|\, \phi_1\rangle \Rightarrow \langle t \,|\, \psi\rangle \quad \dots \quad \langle s_n \,|\, \phi_n\rangle \Rightarrow \langle t \,|\, \psi\rangle}{\langle s \,|\, \phi\rangle \Rightarrow \langle t \,|\, \psi\rangle}\ \text{if } \langle s \,|\, \phi\rangle \text{ is } \mathcal{R}\text{-runnable},$$

where $\Delta_{\mathcal{R}}(\langle s \,|\, \phi\rangle) = \{\langle s_i \,|\, \phi_i\rangle \mid 1 \leq i \leq n\}$ for some $n > 0$.

$$\textsc{Circ}\ \frac{\langle t' \,|\, \phi \wedge (\exists \vec{x}.\ (s = s') \wedge \phi') \wedge \psi'\rangle \Rightarrow \langle t \,|\, \psi\rangle \qquad \langle s \,|\, \phi \wedge \neg(\exists \vec{x}.\ (s = s') \wedge \phi')\rangle \Rightarrow \langle t \,|\, \psi\rangle}{\langle s \,|\, \phi\rangle \Rightarrow \langle t \,|\, \psi\rangle}$$

where $(\langle s' \,|\, \phi'\rangle \Rightarrow \langle t' \,|\, \psi'\rangle) \in G$ and $\{\vec{x}\} = \mathcal{V}ar(s', \phi')$.

Figure 2 Inference rules of DCC [7] for partial validity of APR predicates in G of an LCTRS $\mathcal{R}$ with an underlying theory $\mathcal{T}$.

The two proof systems DVP and DCC have no *one-to-one correspondence* between their inference rules. The main role of Circ in DCC is the introduction of *circularity* to proof trees of DCC so as to make the trees finite as in *cyclic proofs* [3]. Since the initial role of DVP is to define partial validity, DVP does not consider circularity and is used to construct possibly infinite proof trees as in LKID$^\omega$ [4, Chapter 4]. When we do not use Circ in DCC and allow the construction of infinite proof trees, roughly speaking, rules Axiom, Subs, and Der in DCC correspond to rules Subsumption and Step in DVP, while there is no one-to-one correspondence between the inference rules in DCC and DVP. The absence of one-to-one correspondence makes the correctness proof of DCC non-trivial. If DCC were clearly an instance of DVP, then the correctness of DCC would be immediate from that of DVP; it suffices to show that rules in DCC are instances of corresponding rules in DVP. In addition, the abstract framework is sufficiently useful to investigate APR-based approaches to verification of, e.g., *safety* and *liveness* properties, because concurrent transition systems which can be represented by ARSs are usual models for such properties.

In this paper, we aim to develop an abstract foundation for APR analysis toward runtime-error verification. To this end, we revisit the framework for partial validity of APR analysis and adapt it to verification of not only safety but also liveness properties such as race and starvation freedom.

To apply APR analysis to runtime-error verification, a weakened but easily implementable variant of DCC for LCTRSs has been proposed for simpler APR predicates [15]. The simplification goes well with the verification of *safety* properties such as race freedom. On the other hand, it requires the verification of *liveness* properties such as *starvation freedom* to introduce some approximations. For example, a counter for the waiting time to acquire a semaphore has been introduced to ensure starvation freedom [14]: states in which the counter value exceeds an upper limit specified in advance for verification are approximately considered error states.

The APR framework seems to be well-suited to verification of liveness properties: We let P_0 be either the set of initial states or a set of intermediate states to reach states in a target set Q; then, we attempt to prove partial validity of $P_0 \Rightarrow Q$; the difference operation $P \setminus Q$

leaves states in P that have not yet reached any state in Q. The shortcoming of partial validity is that infinite execution paths do not have to be considered, while it does not matter for verification of safety properties: Roughly speaking, reduction sequences ending with error states of safety properties are finite execution paths, and infinite execution paths can be excluded. However, for general runtime-error verification, not only terminating executions but also non-terminating ones must be taken into account, because a runtime error to be verified may happen in a non-terminating execution. For this problem, in [15], simpler APR predicates have been introduced as mentioned above, and a given LCTRS is modified in order to make all finite prefixes of (possibly infinite) execution paths *finite execution paths* of the modified LCTRS. Unfortunately, the limitation of APR predicates eliminates the good compatibility that exists between the difference operation $P \setminus Q$ and liveness verification.

We first reformulate an abstract framework for partial validity of APR predicates w.r.t. ARSs so that there is a one-to-one correspondence between the inference rules for partial validity w.r.t. ARSs and LCTRSs. To be more precise, we reformulate inference rules for partial validity w.r.t. ARSs (Section 3), and adapt a cyclic-proof system for partial validity w.r.t. LCTRSs in [17, Section 3.2] to ARSs (Section 4), providing a formal description of proof trees for partial validity of APR predicates. Secondly, we show how to apply APR analysis to safety verification (Section 5). Thirdly, to apply APR analysis to liveness verification, we introduce a novel, stronger validity of APR problems, called *total validity*, which requires not only finite but also infinite execution paths to reach target sets (Section 6.1). Note that partial validity is a necessary condition for total validity. Finally, for a partially valid APR predicate with a cyclic-proof tree, we show that the acyclicity of the *proof graph* obtained from the cyclic-proof tree is a necessary and sufficient condition for total validity, showing how to apply APR analysis to liveness verification (Section 6.2). All omitted proofs of the claims are provided in the appendix.

As mentioned above, partial validity takes into account finite execution paths only. To apply the APR framework to runtime-error verification, however, *all* (i.e., not only finite but also infinite) execution paths have to be taken into account. To this end, we introduce total validity and show that for a partially valid APR predicate with a proof tree in cyclic-proof style, there is no cycle in the proof graph obtained from the tree if and only if the APR predicate is totally valid w.r.t. $\mathcal{A}$. As a consequence, our sufficient condition for total validity of $P \Rightarrow Q$ w.r.t. $\mathcal{A}$ is the existence of an APR proof, the proof graph of which is acyclic. The cyclic-proof system for APR predicates provides the formal definition of proof graphs, which form the basis for the necessary and sufficient condition of total validity and, in turn, for the reduction of liveness properties to APR predicates.

Since "demonical validity" [7] (partial validity in this paper) does not consider infinite execution paths and we consider a stronger validity for all execution paths, we renamed "demonical validity" to "partial validity", which originates from *partial* and *total* correctness of programs. Notice that total correctness of programs requires termination of programs, but total validity in this paper does not require termination but requires all (possibly infinite) execution paths to reach elements in target sets.

The main contributions of this paper are (i) a reformulation of the APR framework for ARSs, (ii) a novel notion of validity – total validity – for APR predicates, and (iii) a sufficient condition of total validity for, e.g., APR-based liveness verification. As demonstrated in this paper for liveness verification, there must be ample scope for further extending the APR framework. In addition, the APR framework can be ported to other kinds of rewrite systems due to its usefulness in program verification. However, the abstract framework consisting of the two inference rules in DVP is too simple to discuss practical applications alongside several

rewrite systems, such as LCTRSs. Note that two formulations already exist for LCTRSs: the original one [20] and its extension [7]. Therefore, the abstract APR framework formulated in this paper would be highly significant for further research on APR-based verification.

2 Preliminaries

In this section, we briefly recall all-path reachability of ARSs [7]. Familiarity with basic notions and notations on ARSs [1, 24] is assumed.

An *abstract reduction system* (ARS, for short) over an object set A is a pair $(A, \to_{\mathcal{A}})$ such that $\to_{\mathcal{A}}$ is a binary relation over A. The reflexive closure of $\to_{\mathcal{A}}$ is denoted by $\to_{\mathcal{A}}^{=}$, the transitive closure of $\to_{\mathcal{A}}$ is denoted by $\to_{\mathcal{A}}^{+}$, and the reflexive and transitive closure of $\to_{\mathcal{A}}$ is denoted by $\to_{\mathcal{A}}^{*}$. We denote the set of normal forms (irreducible elements) of $\mathcal{A}$ by $NF_{\mathcal{A}}$. An element $s \in A$ is said to *have a normal form* if there exists a normal form $b \in NF_{\mathcal{A}}$ such that $a \to_{\mathcal{A}}^{*} b$. In the remainder of the paper, we use an ARS $\mathcal{A} = (A, \to_{\mathcal{A}})$ without notice.

The *derivative* of a set P ($\subseteq A$) is the set $\partial_{\mathcal{A}}(P) = \{t \in A \mid \exists s \in P.\ s \to_{\mathcal{A}} t\}$. The set P is called *runnable w.r.t.* $\mathcal{A}$ ($\mathcal{A}$-runnable, for short) if $P \neq \emptyset$ and $P \cap NF_{\mathcal{A}} = \emptyset$. By definition, it is clear that $\partial_{\mathcal{A}}(P) \neq \emptyset$ if and only if $P \setminus NF_{\mathcal{A}} \neq \emptyset$. We define execution paths of ARSs.

▶ **Definition 2.1** (execution path [7]). *Let τ be a (possibly infinite) reduction sequence $s_1 \to_{\mathcal{A}} s_2 \to_{\mathcal{A}} \cdots$ of $\mathcal{A}$. We say that τ is an* execution path *(of $\mathcal{A}$) if τ is maximal, i.e., τ is either finite ending with an irreducible element or infinite.*

Note that execution paths of $\mathcal{A}$ are defined coinductively by the inference rules in DVP [7].

Next, we define APR predicates over A.

▶ **Definition 2.2** (APR predicate [7]). *An* all-path reachability *(APR, for short)* predicate *over A is a pair $P \Rightarrow Q$ of $P, Q \subseteq A$.*[1] *Note that Q is not restricted to non-empty sets.*

Note that an APR predicate is defined over a set of objects, independent of any ARSs.

Let *Rules* be a finite set of inference rules over a set $\mathcal{O}$ of objects, which are of the form $\frac{A_1 \ \cdots \ A_n}{A}$ with $A_1, \ldots, A_n \in \mathcal{O}$. $\widehat{Rules}$ stands for the functional of *Rules* (see e.g., [5, Appendix A.2]): For a set X ($\subseteq \mathcal{O}$), $\widehat{Rules}(X) = \{A \mid \frac{A_1 \ \cdots \ A_n}{A} \in Rules, A_1, \ldots, A_n \in X\}$. We denote the greatest fixed point of $\widehat{Rules}$ by $\nu\widehat{Rules}$, i.e., $\nu\widehat{Rules}$ is the greatest set Y ($\subseteq \mathcal{O}$) such that $Y \supseteq \widehat{Rules}(Y)$.

An APR predicate $P \Rightarrow Q$ over A is said to be *partially valid* (or *demonically valid* [7]) w.r.t. $\mathcal{A}$ if each execution path starting from an element in P either eventually reaches an element in Q or is infinite. Partial validity is defined coinductively as follows.

▶ **Definition 2.3** (partial validity [7]). *An APR predicate $P \Rightarrow Q$ over A is said to be* partially valid *w.r.t. $\mathcal{A}$, written as $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$, if $(P \Rightarrow Q) \in \nu\widehat{\mathsf{DVP}}$, where* DVP *consists of the rules* Subsumption *and* Step *in Figure 1. Note that* DVP *implicitly takes $\mathcal{A}$ as a parameter. We write $\mathcal{A} \not\models^{\forall}_{\text{partial}} P \Rightarrow Q$ if $P \Rightarrow Q$ is not partially valid w.r.t. $\mathcal{A}$.*

By definition, APR predicates trivially have the following properties for partial validity.

▶ **Proposition 2.4.** *Let $P, P', Q, Q', R \subseteq A$. Then, all of the following hold:*
(1) *if $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$ and $\mathcal{A} \models^{\forall}_{\text{partial}} P' \Rightarrow Q'$, then $\mathcal{A} \models^{\forall}_{\text{partial}} (P \cup P') \Rightarrow (Q \cup Q')$,*
(2) *if $\mathcal{A} \models^{\forall}_{\text{partial}} (P \cup P') \Rightarrow Q$, then $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$ and $\mathcal{A} \models^{\forall}_{\text{partial}} P' \Rightarrow Q$,*

[1] APR predicates in this paper are called *reachability property* and *reachability formulas* for ARSs and LCTRSs, respectively, in [7], and *APR problems* in [15, 14, 17, 13]. Since we deal with "all-path reachability" only, we only use "all-path reachability predicate" to unify the terminologies.

(3) *if* $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$*, then* $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q'$ *for any* Q' *such that* $Q \subseteq Q' \subseteq A$*,*
(4) *if* $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$ *and* $\mathcal{A} \models^{\forall}_{\text{partial}} Q \Rightarrow R$*, then* $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow R$*, and*
(5) $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow \emptyset$ *if and only if no element in* P *has a normal form (i.e.,* $\{t \in NF_{\mathcal{A}} \mid \exists s \in P.\ s \to^*_{\mathcal{A}} t\} = \emptyset$*).*

For an APR predicate $P \Rightarrow Q$, regarding partial validity, Proposition 2.4 (1)–(2) implies soundness and completeness of splitting P into n sets $P_1, \ldots, P_n$ (i.e., $P = P_1 \cup \cdots \cup P_n$): $\mathcal{A} \models^{\forall}_{\text{partial}} (P_1 \cup \cdots \cup P_n) \Rightarrow Q$ if and only if $\mathcal{A} \models^{\forall}_{\text{partial}} P_i \Rightarrow Q$ for all $1 \leq i \leq n$. On the other hand, splitting the target set Q into $Q_1, \ldots, Q_n$ (i.e., $Q = Q_1 \cup \cdots \cup Q_n$) is sound by Proposition 2.4 (1), but not complete in general: if $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q_i$ for all $1 \leq i \leq n$, then, $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow (Q_1 \cup \cdots \cup Q_n)$, but the other direction does not hold in general (see Example 2.5 below).

The inference rules in DVP can be considered a proof system for partial validity of APR predicates, i.e., $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$ if and only if there exists a (possibly infinite) proof tree obtained from $P \Rightarrow Q$ by applying the rules of DVP.

▶ **Example 2.5.** Consider the ARS $\mathcal{A}_1 = (\{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}\}, \to_{\mathcal{A}_1})$ such that $\to_{\mathcal{A}_1} = \{(\mathsf{a}, \mathsf{b}), (\mathsf{a}, \mathsf{d}), (\mathsf{b}, \mathsf{a}), (\mathsf{b}, \mathsf{c})\}$. We have that $\mathcal{A}_1 \not\models^{\forall}_{\text{partial}} \{\mathsf{a}\} \Rightarrow \{\mathsf{c}\}$ because the finite execution path $\mathsf{a} \to_{\mathcal{A}_1} \mathsf{d}$ does not reach c, and thus $(\{\mathsf{a}\} \Rightarrow \{\mathsf{c}\}) \notin \nu\widehat{\mathsf{DVP}}$ w.r.t. $\mathcal{A}_1$. In fact, only the rule Step is applicable, and we obtain the following stuck incomplete proof tree:

$$\frac{\{\mathsf{b}, \mathsf{d}\} \Rightarrow \{\mathsf{c}\}}{\{\mathsf{a}\} \Rightarrow \{\mathsf{c}\}}\ (\textsc{Step})$$

On the other hand, we have that $\mathcal{A}_1 \models^{\forall}_{\text{partial}} \{\mathsf{a}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\}$ because $(\{\mathsf{a}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\}) \in \nu\widehat{\mathsf{DVP}}$ w.r.t. $\mathcal{A}_1$, i.e., we have the following infinite proof tree:

$$\cfrac{\cfrac{\cfrac{\cfrac{\vdots}{\{\mathsf{b}, \mathsf{d}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\}}\ (\textsc{Step})}{\{\mathsf{a}, \mathsf{c}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\}}\ (\textsc{Step})}{\{\mathsf{b}, \mathsf{d}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\}}\ (\textsc{Step})}{\{\mathsf{a}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\}}\ (\textsc{Step})$$

To prove that $\mathcal{A}_1 \models^{\forall}_{\text{partial}} \{\mathsf{b}, \mathsf{d}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\}$, by Proposition 2.4 (2), we can split $\{\mathsf{b}, \mathsf{d}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\}$ into $\{\mathsf{b}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\}$ and $\{\mathsf{d}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\}$, both of which are partially valid w.r.t. $\mathcal{A}_1$. On the other hand, $\{\mathsf{b}, \mathsf{d}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\}$ cannot be split into $\{\mathsf{b}, \mathsf{d}\} \Rightarrow \{\mathsf{c}\}$ and $\{\mathsf{b}, \mathsf{d}\} \Rightarrow \{\mathsf{d}\}$, the former of which is not partially valid w.r.t. $\mathcal{A}_1$.

3 Reformulation of Inference Rules for ARSs

In this section, we reformulate the proof system DVP for partial validity w.r.t. ARSs to establish a one-to-one correspondence between the inference rules for ARSs and LCTRSs.

By abstracting the inference rules in DCC for LCTRSs in Figure 2, we obtain inference rules for ARSs, which have one-to-one correspondence with the inference rules for LCTRSs.

▶ **Definition 3.1** (DVP_+)**.** *We define* DVP_+ *consisting of the inference rules in Figure 3.*

In applying Subs and Der of DVP_+ to $P \Rightarrow Q$, we can freely split the results of $P \setminus Q$ and $\partial_{\mathcal{A}}(P)$ into n sets $P_1, \ldots, P_n$, respectively. If $P_i = \emptyset$ for Subs or Der, then the generated subgoal $P_i \Rightarrow Q$ can immediately be proved by Axiom. To avoid such redundant splits, we assume that

$$\textsc{Axiom}\ \frac{}{P \Rightarrow Q}\ \text{if } P = \emptyset. \qquad \textsc{Subs}\ \frac{P_1 \Rightarrow Q \quad \dots \quad P_n \Rightarrow Q}{P \Rightarrow Q}\ \text{if } P \cap Q \neq \emptyset,$$

where $P \setminus Q = P_1 \cup \cdots \cup P_n$ for some $n > 0$.

$$\textsc{Der}\ \frac{P_1 \Rightarrow Q \quad \dots \quad P_n \Rightarrow Q}{P \Rightarrow Q}\ \text{if } P \cap Q = \emptyset \text{ and } P \text{ is } \mathcal{A}\text{-runnable},$$

where $\partial_{\mathcal{A}}(P) = P_1 \cup \cdots \cup P_n$ for some $n > 0$.

Figure 3 Rules of DVP_+ for proving partial validity of APR predicates of an ARS $\mathcal{A}$.

- "if $n > 1$ then $\emptyset \subset P_i \subseteq P$ for all $1 \leq i \leq n$" for Subs, and
- "$P_i \neq \emptyset$ for all $1 \leq i \leq n$" for Der.

Note that for an $\mathcal{A}$-runnable set P, we have that $\partial_{\mathcal{A}}(P) \neq \emptyset$ for Der, because $P \setminus NF_{\mathcal{A}} \neq \emptyset$. Note also that P_i and P_j $(i \neq j)$ may overlap, i.e., $P_i \cap P_j \neq \emptyset$.

By definition, the side conditions of rules in DVP_+ are orthogonal, and thus, for each APR predicate, at most one rule in DVP_+ is applicable (cf. [17]).

▶ **Proposition 3.2.** *Let $P, Q \subseteq A$. Then, at most one rule in DVP_+ is applicable to the APR predicate $P \Rightarrow Q$.*

Note that the way of applying Subs and Der in DVP_+ is not unique, because multiple decompositions of resulting subgoals are possible.

For any ARS, DVP and DVP_+ coincide.

▶ **Theorem 3.3.** *For any ARS $\mathcal{A}$, $\nu\widehat{\mathsf{DVP}} = \nu\widehat{\mathsf{DVP}_+}$.*

By Theorem 3.3, DVP and DVP_+ define the same partially valid APR predicates and they are equivalent as proof systems for partial validity of APR predicates. On the other hand, DVP_+ has more flexibility than DVP w.r.t. the split of source sets of generated subgoals, in addition to the one-to-one correspondence with DCC in terms of description.

The disproof criterion for LCTRSs [17] is formulated for ARSs as follows.

▶ **Proposition 3.4.** *Let $P \Rightarrow Q$ be an APR predicate over A. If $P \cap Q = \emptyset$ and $P \cap NF_{\mathcal{A}} \neq \emptyset$, then $\mathcal{A} \not\models^{\forall}_{\text{partial}} P \Rightarrow Q$.*

Proof. Assume that $P \cap Q = \emptyset$ and $P \cap NF_{\mathcal{A}} \neq \emptyset$. Then, there exists a normal form $s \in P \cap NF_{\mathcal{A}}$ such that $s \notin Q$, and thus we have a finite execution path s that does not reach Q. Therefore, we have that $\mathcal{A} \not\models^{\forall}_{\text{partial}} P \Rightarrow Q$. ◀

By definition, it is clear that $P \neq \emptyset$ and $P \cap NF_{\mathcal{A}} \neq \emptyset$ if and only if $P \neq \emptyset$ and P is not $\mathcal{A}$-runnable. Proposition 3.4 implies an inference rule for disproof (cf. [17]).

▶ **Definition 3.5** ($\mathsf{DVP}_\bot$)**.** *We define $\mathsf{DVP}_\bot$ consisting of the inference rules in DVP_+ and the following rule for disproof:*

$$\textsc{Dis}\ \frac{\bot}{P \Rightarrow Q}\ \textit{if } P \cap Q = \emptyset,\ P \neq \emptyset, \textit{ and } P \textit{ is not } \mathcal{A}\textit{-runnable.}$$

Note that $\bot$ is used as a special case – the invalid *one – of APR predicates.*

By definition, the side conditions of rules in $\mathsf{DVP}_\bot$ are orthogonal and exhaustive, and thus, for each APR predicate, exactly one rule in $\mathsf{DVP}_\bot$ is applicable (cf. [17]).

▶ **Proposition 3.6.** *Let $P, Q \subseteq A$. Then, exactly one rule in $\mathsf{DVP}_\bot$ is applicable to the APR predicate $P \Rightarrow Q$.*

4 Cyclic Proof System for APR Problems of ARSs

In this section, we adapt the cyclic-proof system for partial validity of APR predicates w.r.t. LCTRSs [17] to the abstract APR framework reformulated in Section 3. To this end, we first revisit Circ in Figure 2, and then define APR pre-proofs for partial validity w.r.t. ARSs.

In proving partial validity of APR predicates using the inference rules such as $\mathsf{DVP}_\bot$, we would like to construct finite proof trees if possible. In the APR framework, a *circularity* rule, Circ, has been introduced [7, Definition 12] (see Figure 2). Roughly speaking, the rule can be considered as a composition of "circularity" in *cyclic proofs* [3] and "cut" in *sequent calculus*. Rule Circ in Figure 2 for LCTRSs is formulated for ARSs as follows:

$$\textsc{Circ}\ \frac{Q' \Rightarrow Q \qquad (P \setminus P') \Rightarrow Q}{P \Rightarrow Q}\ \text{if } (P' \Rightarrow Q') \in G,$$

where G is a given set of APR predicates over A. Note that $P' \Rightarrow Q'$ may be the same as $P \Rightarrow Q$, while Der has to be applied to $P' \Rightarrow Q'$ somewhere for soundness. Note also that Q' of the subgoal $Q' \Rightarrow Q$ may be improved by replacing it with Q'' such that $\{t \in Q' \mid \exists s \in P \cap P'.\ s \to_{\mathcal{A}}^* t\} \subseteq Q'' \subseteq Q'$. As stated in Section 1, G is assumed to be given in advance as a set of APR predicates to be proved partially valid. In practice, we start with a single APR predicate to be proved partially valid, G is the empty set; when we apply Der to an APR predicate, we add the predicates to G; for soundness, for each APR predicate to which Circ is applied, there must exist an APR predicate in G to which Der has already been applied, i.e., Der must be applied to all APR predicates in G.

From the viewpoint of practical use, we do not consider "cut" because, as for the use of "cut" in proof theories, it is not so easy to split an APR predicate $P \Rightarrow Q$ into appropriate ones $P \Rightarrow R$ and $R \Rightarrow Q$ for some set R; deriving R is similar to a human deriving a lemma. Then, the circularity rule without "cut" is formulated for ARSs as follows:

$$\textsc{Cyc}\ \frac{}{P \Rightarrow Q}\ \text{if } P \subseteq P' \text{ and } Q \supseteq Q' \text{ for some } (P' \Rightarrow Q') \in G,$$

where G is a given set of APR predicates over A. To distinguish the circularity rule without "cut" from Circ, we named the former Cyc, which originates from "cyclic proofs". The role of Cyc is not only "circularity" but also "generalization" of APR predicates: $P' \Rightarrow Q'$ is more general than $P \Rightarrow Q$ in the sense that if $\mathcal{A} \models^{\forall}_{\text{partial}} P' \Rightarrow Q'$, then $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$ (Proposition 2.4 (1)–(2)). Viewed in this light, if $P = P'$ and $Q = Q'$, then Cyc is just "circularity" in cyclic proofs, and otherwise, Cyc generalizes $P \Rightarrow Q$ to $P' \Rightarrow Q'$, which is included in G to be proved partially valid as, e.g., a more general subgoal for the main goal. Note that "cut" is formulated as follows:

$$\textsc{Cut}\ \frac{P \Rightarrow Q' \qquad Q' \Rightarrow Q}{P \Rightarrow Q}$$

As in other proof systems, Cut must be powerful but needs a heuristic for automation. As a first step, we leave the introduction of Cut to the proof system below as future work.

In [17], a simpler proof system for partial validity w.r.t. LCTRSs has been formulated in the *cyclic-proof* style [3]. In the proof system, the circularity rule is not explicitly used, but the *bud-companion* relationship of cyclic proofs is used instead in proof trees. We formulate the cyclic-proof system for partial validity w.r.t. ARSs.

▶ **Definition 4.1** (derivation tree of DVP). *An* APR derivation tree *w.r.t.* $\mathcal{A} = (A, \to_{\mathcal{A}})$ *is a finite tree* $\mathcal{T} = (V, \mathrm{a}, \mathrm{r}, c)$ *such that*

- V *is a finite set of nodes,*

Table 1 How mappings a, r, c, and ξ are defined for a node $v \in V$ regarding an APR pre-proof $((V, a, r, c), \xi)$.

$a(v)$	$r(v)$	$c(v)$	open/closed	ξ
$P \Rightarrow Q$	AXIOM	ϵ	closed leaf	(not in the domain of ξ)
	SUBS	$v_1 \dots v_n$ $(n > 0)$	(internal node)	
	DER			
	DIS	v_1 such that $a(v_1) = \bot$		
	undefined	undefined	open leaf	defined or undefined
$\bot$	undefined	undefined	closed leaf	(not in the domain of ξ)

- *a is a total mapping from V to the set of APR predicates over A, which includes $\bot$ as an APR predicate,*
- *r is a partial mapping from V to $\mathsf{DVP}_\bot$,*
- *c is a partial mapping from V to V^* (we write $c_j(v)$ for the j-th component of $c(v)$) which is the j-th child of v, and*
- *for all nodes $v \in V$, $c_j(v)$ is defined just in case $r(v)$ is a rule with k premises ($1 \le j \le k$), and $\frac{a(c_1(v)) \ \dots \ a(c_k(v))}{a(v)}$ is an instance of rule $r(v)$, where if $k = 0$, then $c(v) = \epsilon$.*

Note that a node $v \in V$ is a leaf if and only if either $c(v) = \epsilon$ or $c(v)$ is undefined. For a node v with $a(v) = (P \Rightarrow Q)$, $a_L(v)$ and $a_R(v)$ denote P and Q, respectively. A leaf v of $\mathcal{T}$ is said to be closed *if $c(v) = \epsilon$ or $a(v) = \bot$. A leaf v of $\mathcal{T}$ is said to be* open *if it is not closed, i.e., $c(v) \neq \epsilon$ and $a(v) \neq \bot$. The set of open leaves of $\mathcal{T}$ is denoted by V_{open}.*

Note that any node v with $a(v) \neq \bot$ must have a rule attached, i.e., $r(v)$ must be defined. By introducing the circularity relationship into APR derivation trees, we define APR pre-proofs.

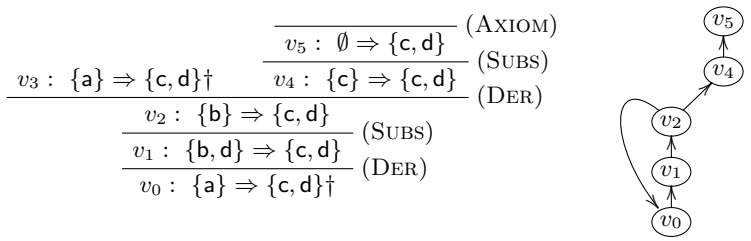
▶ **Definition 4.2** (APR pre-proof). *An* APR pre-proof *for an APR predicate $P \Rightarrow Q$ w.r.t. $\mathcal{A}$ is a pair $(\mathcal{T}, \xi)$ of an APR derivation tree $\mathcal{T} = (V, a, r, c)$ (with v_0 the root node) and a partial mapping ξ from V_{open} to $V \setminus V_{open}$ such that $a(v_0) = (P \Rightarrow Q)$, and for any open leaf v, if $\xi(v)$ is defined, then $\xi(v)$ is a node of $\mathcal{T}$ such that $a(v) = a(\xi(v))$[2] and $r(\xi(v)) = \text{DER}$. An open leaf v with $\xi(v)$ defined is called a* bud *node of $\mathcal{T}$, and the node $\xi(v)$ is called a* companion *of v. We denote the set of bud nodes in V by V_{bud} ($\subseteq V_{open}$). The APR pre-proof $(\mathcal{T}, \xi)$ is said to be* closed *if any leaf $v \in V$ is either closed or a bud node. The APR pre-proof $(\mathcal{T}, \xi)$ is said to be* open *if $\mathcal{T}$ is not closed, i.e., there exists an open leaf that is not a bud.*

Note that a companion does not have to be an ancestor of its bud nodes. Table 1 illustrates how a, r, c, and ξ are defined for a node $v \in V$ regarding an APR pre-proof $((V, a, r, c), \xi)$. Unlike the usual definition of companions in cyclic-proofs, for a bud node v, we required the additional condition "$r(\xi(v)) = \text{DER}$". The reason for requiring the additional condition will be explained later.

▶ **Example 4.3.** Let us consider the ARS $\mathcal{A}_1$ in Example 2.5 again. Let $v_0, v_1, \dots, v_5$ be nodes such that

- $a(v_0) = (\{\mathsf{a}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\})$, $r(v_0) = \text{DER}$, $c(v_0) = v_1$,
- $a(v_1) = (\{\mathsf{b}, \mathsf{d}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\})$, $r(v_1) = \text{SUBS}$, $c(v_1) = v_2$,

[2] From the viewpoint of CYC, the condition "$a(v) = a(\xi(v))$" can be relaxed to "$a_L(v) \subseteq a_L(\xi(v))$". However, as a starting point, we do not use the relaxed condition and leave it as future work.



(a) APR pre-proof (b) proof graph

Figure 4 An APR pre-proof and its proof graph for the APR predicate $\{\mathsf{a}\} \Rightarrow \{\mathsf{c},\mathsf{d}\}$, where † indicates the bud-companion relationship.

- $a(v_2) = (\{\mathsf{b}\} \Rightarrow \{\mathsf{c},\mathsf{d}\})$, $r(v_2) = \textsc{Der}$, $c(v_2) = v_3 v_4$,
- $a(v_3) = (\{\mathsf{a}\} \Rightarrow \{\mathsf{c},\mathsf{d}\})$, neither $r(v_3)$ nor $c(v_3)$ is defined,
- $a(v_4) = (\{\mathsf{c}\} \Rightarrow \{\mathsf{c},\mathsf{d}\})$, $r(v_4) = \textsc{Subs}$, $c(v_4) = v_5$,
- $a(v_5) = (\emptyset \Rightarrow \{\mathsf{c},\mathsf{d}\})$, $r(v_5) = \textsc{Axiom}$, $c(v_5) = \epsilon$, and
- $\xi(v_3) = v_0$.

Then, $((\{v_0, v_1, \ldots, v_5\}, \mathrm{a}, r, c), \xi)$ is an APR pre-proof for $\{\mathsf{a}\} \Rightarrow \{\mathsf{c},\mathsf{d}\}$ w.r.t. $\mathcal{A}_1$, which is visualized in Figure 4 (a).

The proof graph obtained from a closed APR pre-proof is defined as follows.

▶ **Definition 4.4** (proof graph). *The* proof graph *of a closed APR pre-proof $(\mathcal{T}, \xi)$ with $\mathcal{T} = (V, \mathrm{a}, r, c)$ is a directed graph (V', E) obtained from $\mathcal{T}$ by identifying each bud node and its companion:*

- *$V' = V \setminus V_{open}$, where each node in V is associated with the APR predicate $\mathrm{a}(v)$, and*
- *$(v, v') \in E$ if and only if there exists a node $v'' \in V$ such that v'' appears in $c(v)$, if v'' is a bud node then v' is a companion of v'', and otherwise, $v'' = v'$.*

For an edge $(v, v') \in E$ and a rule name $X \in \{\textsc{Axiom}, \textsc{Subs}, \textsc{Der}, \textsc{Dis}\}$, we write $v \to_X v'$ if $r(v) = X$.

▶ **Example 4.5.** The proof graph of $((\{v_0, v_1, \ldots, v_5\}, \mathrm{a}, r, c), \xi)$ in Example 4.3 (i.e., Figure 4 (a)) is illustrated in Figure 4 (b).

By definition, nodes of proof graphs trivially have the following properties.

▶ **Proposition 4.6.** *Let $(\mathcal{T}, \xi)$ be a closed APR pre-proof, $\mathcal{T} = ((V, \mathrm{a}, r, c), (V, E)$ be the proof graph of $(\mathcal{T}, \xi)$, $v, v' \in V$, and $X \in \{\textsc{Axiom}, \textsc{Subs}, \textsc{Der}, \textsc{Dis}\}$. Then, all of the following statements hold:*

(1) *if $(v, v') \in E$, then $\mathrm{a}_R(v) = \mathrm{a}_R(v')$,*
(2) *if $v \to_{\textsc{Subs}} v'$, then $(\mathrm{a}_L(v) \setminus \mathrm{a}_R(v)) \supseteq \mathrm{a}_L(v')$,*
(3) *if $v \to_{\textsc{Der}} v'$, then both of the following hold:*
  - *for any $s \in \mathrm{a}_L(v)$, there exist a node $v'' \in V$ and an element $t \in \mathrm{a}_L(v'')$ such that $v \to_{\textsc{Der}} v''$ and $s \to_{\mathcal{A}} t$, and*
  - *for any $t \in \mathrm{a}_L(v')$, there exists an element $s \in \mathrm{a}_L(v)$ such that $s \to_{\mathcal{A}} t$,*

(4) *if $v \to_{\textsc{Dis}} v'$, then $\mathrm{a}(v') = \bot$, and*
(5) *if $r(v) = \textsc{Axiom}$, then $\mathrm{a}_L(v) = \emptyset$ and v has no outedge.*

Cyclic proofs satisfy the *global trace condition* [3]. The application of rules of *inductive predicates* is the measure of coinduction in the cyclic-proof setting, which ensures the soundness of cyclic proofs. Roughly speaking, the global trace condition requires every

$$\dfrac{\dfrac{\bot}{\{\mathsf{b},\mathsf{d}\} \Rightarrow \{\mathsf{c}\}}\,(\textsc{Dis})}{\{\mathsf{a}\} \Rightarrow \{\mathsf{c}\}}\,(\textsc{Der})$$

Figure 5 An APR pre-proof for $\{\mathsf{a}\} \Rightarrow \{\mathsf{c}\}$, which is an APR disproof.

infinite *trace* – a sequence of atomic formulas in premises sets of sequents – to follow an infinite path of the proof graph, which goes through infinitely many edges corresponding to the application of rules of inductive predicates. In our case, such edges correspond to those from nodes with Der and for soundness of an APR proof, the APR proof has to satisfy the global trace condition, i.e., every infinite path of the proof graph has to go through node with Der infinitely often. To make proof graphs implicitly satisfy the global trace condition, we require companions to be nodes with Der; the form of APR predicates can be considered the same as that of sequents, and each APR predicate has exactly one set corresponding to an atomic formula of the premises of sequents; viewed in this light, traces correspond to paths of proof graphs of closed APR pre-proofs; every cycle goes through companions infinitely often, and thus every infinite trace follows an infinite path that goes through infinitely many nodes with Der.

The requirement to the bud-companion relationship – companions have Der attached – does not lose generality. Let $((V, a, r, c), \xi)$ be an APR pre-proof for an APR predicate $P \Rightarrow Q$ w.r.t. $\mathcal{A}$, v be a bud node, and v' be a companion of v (i.e., $\xi(v) = v'$). If $r(v') \neq \textsc{Der}$, then we can drop the bud-companion relationship between v and v', obtaining another APR pre-proof for $P \Rightarrow Q$ as follows:

- if $r(v') \in \{\textsc{Axiom}, \textsc{Dis}\}$, then we let $r(v) = r(v')$ and $c(v) = c(v')$, and
- if $r(v') = \textsc{Subs}$ and $c(v') = v'_1 \dots v'_n$, then we introduce new n nodes $v_1, \dots, v_n$ to V and let $r(v) = \textsc{Subs}$, $c(v) = v_1 \dots v_n$, and $a(v_i) = a(v'_i)$ and $\xi(v_i) = v'_i$ for all $i \in \{1, \dots, n\}$.

While the second case above introduces a new bud-companion relationship to the transformed APR pre-proof, the repetition of the above transformation halts because v'_i is a child of v' such that $r(v) = \textsc{Subs}$ and thus $r(v'_i) \in \{\textsc{Axiom}, \textsc{Der}, \textsc{Dis}\}$.

We now define APR proofs and disproofs as APR pre-proofs satisfying certain conditions.

▶ **Definition 4.7** (APR proof and disproof). *An APR pre-proof $((V, a, r, p), \xi)$ for an APR predicate $P \Rightarrow Q$ w.r.t. $\mathcal{A}$ is called an* APR proof *if the domain of ξ is V_{open} (i.e., all open nodes are bud nodes) and there is no node $v \in V$ such that $r(v) = \textsc{Dis}$. The APR pre-proof $((V, a, r, p), \xi)$ is called an* APR disproof *if there exists a node $v \in V$ such that $r(v) = \textsc{Dis}$.*

Note that a closed APR pre-proof is either an APR proof or an APR disproof. Note also that an APR disproof may have open nodes that are not bud nodes.

▶ **Example 4.8.** The APR pre-proof in Figure 4 (a) is an APR proof for $\{\mathsf{a}\} \Rightarrow \{\mathsf{c}, \mathsf{d}\}$. The APR pre-proof in Figure 5 is an APR disproof for $\{\mathsf{a}\} \Rightarrow \{\mathsf{c}\}$.

APR proofs and disproofs are sound.

▶ **Theorem 4.9.** *Let $P \Rightarrow Q$ be an APR predicate over A. Then, both of the following statements hold:*

(1) *if there exists an APR proof for $P \Rightarrow Q$ w.r.t. $\mathcal{A}$, then $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$, and*

(2) *if there exists an APR disproof of $P \Rightarrow Q$ w.r.t. $\mathcal{A}$, then $\mathcal{A} \not\models^{\forall}_{\text{partial}} P \Rightarrow Q$.*

When $\{t \in A \mid \exists s \in P.\ s \to_{\mathcal{A}}^* t\}$ is finite, the APR predicate $P \Rightarrow Q$ has either an APR proof or an APR disproof. In addition, if the side conditions of rules in $\mathsf{DVP}_\bot$ are decidable, then partial validity of $P \Rightarrow Q$ w.r.t. $\mathcal{A}$ is decidable.

▶ **Theorem 4.10.** *Let $P \Rightarrow Q$ be an APR predicate over A, and $\partial_{\mathcal{A}}^*(P) = \{t \in A \mid \exists s \in P.\ s \to_{\mathcal{A}}^* t\}$. Then, both of the following statements hold:*
(1) *If $\partial_{\mathcal{A}}^*(P)$ is finite, then there exists either an APR proof or disproof for $P \Rightarrow Q$, and*
(2) *if $\partial_{\mathcal{A}}^*(P)$ is finite and the emptiness, intersection emptiness, and $\mathcal{A}$-runnability problems for A are decidable, then partial validity of $P \Rightarrow Q$ w.r.t. $\mathcal{A}$ is decidable.*

When we simultaneously prove two or more APR predicates, i.e., a set G of APR predicates, to be partially valid, the notion of APR derivation trees and pre-proofs can be extended to forests: We consider a set of APR pre-proofs for APR predicates in G; the bud-companion relationship is allowed between nodes in different derivation trees.

The proof system DCC for LCTRSs [7] and its weakened variant [15] are instances of DVP_+ and $\mathsf{DVP}_\bot$, respectively, and APR pre-proofs w.r.t. ARSs, which provide a concrete method for constructing APR (dis)proofs for APR predicates w.r.t. LCTRSs, are used as a foundation for both systems. For the page limitation, in the rest of this section, we give an informal proof for DCC being an instance of DVP_+. A formal proof can be seen in the full version [18] of this paper.

Let $\mathcal{R}$ be an LCTRS over a signature Σ with an underlying theory $\mathcal{T}$ [7, Section 3]. LCTRS $\mathcal{R}$ induces the ARS $(T(\Sigma), \to_{\mathcal{R}})$. A *constrained term* $\langle s \mid \phi \rangle$ consisting of a term s and a constraint ϕ represents the set of ground normalized instances $s\gamma$ w.r.t. ϕ, where γ is a ground normalized substitution and $\phi\gamma$ is evaluated to true. In light of this, we deal with constrained terms as the set of ground instances, and pairs of constrained terms can be considered APR problems over $T(\Sigma)$. We show that each rule in DCC is an instance of the corresponding rule in DVP_+ w.r.t. the ARS $(T(\Sigma), \to_{\mathcal{R}})$. Let us consider an APR problem $\langle s \mid \phi \rangle \Rightarrow \langle t \mid \psi \rangle$.

- By definition, it is clear that ϕ is unsatisfiable if and only if $\langle s \mid \phi \rangle$ is the emptyset. Thus, Axiom in DCC is an instance of Axiom in DVP_+.
- For a constrained rewrite rule $\ell \to r\ [\varphi]$ in $\mathcal{R}$, satisfiability of $(s = t) \wedge \phi \wedge \psi$ means that $\langle s \mid \phi \rangle \cap \langle t \mid \psi \rangle \neq \emptyset$, and the constrained term $\langle s \mid \phi \wedge \neg(\exists \vec{x}.\ ((s = t) \wedge \psi)) \rangle$ represents the set $\langle s \mid \phi \rangle \setminus \langle t \mid \psi \rangle$, where $\{\vec{x}\} = \mathcal{V}ar(t, \psi) \setminus (s, \phi)$. Thus, Subs in DCC is an instance of Subs in DVP_+.
- We have that $\Delta_{\mathcal{R}}(\langle s \mid \phi \rangle) = \partial_{(T(\Sigma), \to_{\mathcal{R}})}(\langle s \mid \phi \rangle)$ [7, Theorem 1]. By definition, $\mathcal{R}$-runnability of $\langle s \mid \phi \rangle$ is defined by $(T(\Sigma), \to_{\mathcal{R}})$-runnability of the set $\langle s \mid \phi \rangle$. Thus, Der in DCC is an instance of Der in DVP_+.
- To remove the "cut" function from Circ in DCC, both "$\langle t' \mid \psi' \rangle \subseteq \langle t \mid \psi \rangle$" and "$\langle s \mid \phi \rangle \subseteq \langle s' \mid \phi' \rangle$" are necessary. Under these conditions, the premises are partially valid and can thus be dropped. Then, a simplified variant without "cut" is obtained as follows:

$$\text{Cyc}\ \frac{}{\langle s \mid \phi \rangle \Rightarrow \langle t \mid \psi \rangle}\ \text{if } \langle s \mid \phi \rangle \subseteq \langle s' \mid \phi' \rangle \text{ and } \langle t \mid \psi \rangle \supseteq \langle t' \mid \psi' \rangle \text{ for some } (\langle s' \mid \phi' \rangle \Rightarrow \langle t' \mid \psi' \rangle) \in G.$$

By definition, the above Cyc is an instance of Cyc in DVP_+.

5 Reduction of Safety Properties to APR predicates

Informally speaking, *safety properties* specify that "something bad never happens" [2, Section 3.3.2]. As a safety property w.r.t. a system and a set E of error states, we consider the property that any (possibly non-terminating) execution of the system never reaches any error state in E. This kind of safety property is formulated as a problem for ARSs as follows:

Instance An ARS $\mathcal{A} = (A, \to_{\mathcal{A}})$ and sets P, E $(\subseteq A)$

Question Is there no (possibly infinite) execution path $s_0 \to_{\mathcal{A}} s_1 \to_{\mathcal{A}} \cdots$ such that $s_0 \in P$ and $s_i \in E$ for some $i \geq 0$?

Note that not all error states are, in general, irreducible. This problem is the same as *non-reachability* from P to E w.r.t. $\mathcal{A}$. On the other hand, this problem can be reduced to the APR problem [15]. In this section, we revisit the reduction of the non-reachability problem to partial validity of APR predicates w.r.t. LCTRSs, and then reformulate it for ARSs.

Note that in general, the negation of partial validity of the APR predicate $P \Rightarrow E$ is not equivalent to non-reachability from P to E:

- The negation of partial validity of $P \Rightarrow E$ is that there exists a finite execution path starting from a state in P, which does not include any error state in E, and
- non-reachability from P to E is that there exists *no* execution path starting from a state in P, which does not include any error state in E.

We first revisit the reduction to APR predicates w.r.t. LCTRSs in [17]. For an LCTRS $\mathcal{R}$ w.r.t. error states represented by constrained terms $\langle e_1 \,|\, \psi_1 \rangle, \ldots, \langle e_k \,|\, \psi_k \rangle$, the reduction proceeds as follows:

1. Fresh constants $\mathsf{success}$ and error are introduced to the signature of $\mathcal{R}$.
2. Constrained rewrite rules to reduce any state, including an error state, to $\mathsf{success}$ are added to $\mathcal{R}$.
3. Constrained rewrite rules $e_i \to \mathsf{error}\ [\psi_i]$ with $1 \leq i \leq k$ are added to $\mathcal{R}$.
4. The non-reachability problem w.r.t. the error states E is reduced to the APR predicate $I \Rightarrow \{\mathsf{success}\}$, where I is the set of ground terms for initial states of a system.

Note that the introduced constants are normal forms of the modified LCTRS. The role of $\mathsf{success}$ is to make all prefixes of any reduction sequence from an initial state a finite execution path ending with $\mathsf{success}$. If an initial execution path reaches an error state represented by some $\langle e_i \,|\, \psi_i \rangle$, then there exists a finite execution path ending with error.

We now adapt the above approach to ARSs. Usually, error states are irreducible, and thus, we consider only irreducible error states. If an error state is reducible, then, as in the approach above for LCTRSs, we introduce a fresh constant such as error, together with the additional reduction from the error states to error, which is considered a dummy error state in the modified system.

Let us reconsider to reduce non-reachability (from P to E w.r.t. an ARS $\mathcal{A}$) to a partially valid APR predicate $P \Rightarrow Q$.[3] For the reduced partially valid APR predicate, we do not have to take into account infinite execution paths, because any execution path ending with an error state is a finite execution path. If $Q \cap E \neq \emptyset$, then the reduction is not sound, and thus Q should have no error state in E: $Q \cap E = \emptyset$. For the partial validity, all normal forms that are reachable from P and are not error states should be included in Q: $\{t \in NF_{\mathcal{A}} \setminus E \mid \exists s \in P.\ s \to_{\mathcal{A}}^* t\} \subseteq Q$. If Q includes a reducible state, then the reduction may not be sound: An error state may be reachable from the reducible state. Thus, Q should be a set of normal forms w.r.t. $\mathcal{A}$. We call such $P \Rightarrow Q$ a *safety APR predicate for E w.r.t. $\mathcal{A}$*.

[3] We do not consider to reduce to APR predicates of the form $P' \Rightarrow Q$ such that $P' \neq P$, because we should consider all execution paths starting from P, and thus the APR predicate $P \Rightarrow P''$ for some $P'' \subseteq P'$ needs to be partially valid w.r.t. $\mathcal{A}$. When we choose an APR predicate of the form $P \Rightarrow Q$, we do not have to find appropriate sets P', P''.

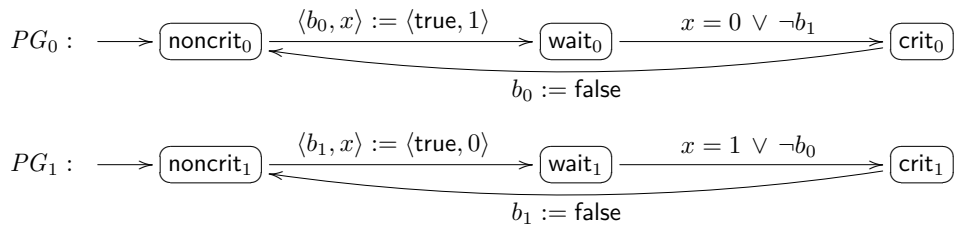


Figure 6 Program graph PG_i $(i = 0, 1)$ for Peterson's mutual exclusion algorithm.

▶ **Definition 5.1** (safety APR predicate for error states). *Let $E \subseteq NF_{\mathcal{A}}$ be a set of error states. An APR predicate $P \Rightarrow Q$ is called a* safety predicate for E w.r.t. $\mathcal{A}$ *if $Q \cap E = \emptyset$, $\{t \in NF_{\mathcal{A}} \setminus E \mid \exists s \in P.\ s \to^*_{\mathcal{A}} t\} \subseteq Q$, and $Q \subseteq NF_{\mathcal{A}}$.*

Safety APR predicates can be used for verification of safety properties.

▶ **Theorem 5.2.** *Let $P, Q \subseteq A$, $E \subseteq NF_{\mathcal{A}}$, and $P \Rightarrow Q$ be a safety APR predicate for E w.r.t. $\mathcal{A}$. Then, $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$ if and only if there is no finite execution path of $\mathcal{A}$ that starts with an element in P and includes an element in E.*

▶ **Example 5.3.** Let us consider the *program graphs* PG_0, PG_1 in Figure 6 for *Peterson's mutual exclusion algorithm* [2, Example 2.25]. Two processes P_0, P_1 specified by PG_0, PG_1, respectively, share Boolean variables b_0, b_1 and an integer variable x; the Boolean variables b_i indicates that P_i wants to enter the critical section crit_i; the integer variable x stores the identifier of the process that has priority for the critical section at that time. A state of the *asynchronous integer transition system* (AITS, for short) with shared variables [2] consisting of PG_0 and PG_1 is a tuple of a location of PG_0, a location of PG_1, and an *assignment* from variables b_0, b_1, x to values. The initial states of the AITS are $\langle\, \mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, m \,\rangle$ with $m \in \{0, 1\}$. Let $State_i$ be the set $\{\mathsf{noncrit}_i, \mathsf{wait}_i, \mathsf{crit}_i\}$ $(i \in \{0, 1\})$, $\mathbb{B}$ be the set $\{\mathsf{true}, \mathsf{false}\}$, and $\mathcal{S}$ be the set of assignments for variables $b_0, b_1 : bool$ and $x : int$, i.e., $\mathcal{S} = \{\{b_0 \mapsto v_0, b_1 \mapsto v_1, x \mapsto m\} \mid v_0, v_1 \in \mathbb{B},\ m \in \mathbb{Z}\}$. We denote a state $\langle\, p_0, p_1, \sigma \,\rangle$ by $\langle\, p_0, p_1, \sigma(b_0), \sigma(b_1), \sigma(x) \,\rangle$, where $\sigma \in \mathcal{S}$. The AITS consisting of P_0 and P_1 is represented by the following ARS:

$$\mathcal{A}_2 = (\{\langle\, p_0, p_1, \sigma \,\rangle \mid p_0 \in State_0,\ p_1 \in State_1,\ \sigma \in \mathcal{S}\}, \to_{\mathcal{A}_2})$$

such that

- $\langle\, \mathsf{noncrit}_0, p_1, v_0, v_1, m \,\rangle \to_{\mathcal{A}_2} \langle\, \mathsf{wait}_0, p_1, \mathsf{true}, v_1, 1 \,\rangle$,
- $\langle\, \mathsf{wait}_0, p_1, v_0, v_1, m \,\rangle \to_{\mathcal{A}_2} \langle\, \mathsf{crit}_0, p_1, \mathsf{true}, v_1, m \,\rangle$ if $m = 0$ or $v_1 = \mathsf{false}$,
- $\langle\, p_0, \mathsf{crit}_1, v_0, v_1, m \,\rangle \to_{\mathcal{A}_2} \langle\, p_0, \mathsf{noncrit}_1, v, \mathsf{false}, m \,\rangle$,
- $\langle\, p_0, \mathsf{noncrit}_1, v_0, v_1, m \,\rangle \to_{\mathcal{A}_2} \langle\, p_0, \mathsf{wait}_1, v_0, \mathsf{true}, 0 \,\rangle$,
- $\langle\, p_0, \mathsf{wait}_1, v_0, v_1, m \,\rangle \to_{\mathcal{A}_2} \langle\, p_0, \mathsf{crit}_1, v_0, \mathsf{true}, m \,\rangle$ if $m = 1$ or $v_0 = \mathsf{false}$, and
- $\langle\, p_0, \mathsf{crit}_1, v_0, v_1, m \,\rangle \to_{\mathcal{A}_2} \langle\, p_0, \mathsf{noncrit}_1, v_0, \mathsf{false}, m \,\rangle$,

where $p_i \in State_i$ with $i \in \{0, 1\}$, $v_0, v_1 \in \mathbb{B}$, and $m \in \mathbb{Z}$.

Let us consider the *race freedom* of mutual exclusion – the two processes do not enter their critical sections simultaneously – for the AITS $\mathcal{A}_2$. There is no normal form reachable from an initial state. This property will be examined later, together with race freedom. The error states are $\langle\, \mathsf{crit}_0, \mathsf{crit}_1, \sigma \,\rangle$ with $\sigma \in \mathcal{S}$. Since all the error states are reducible w.r.t. $\mathcal{A}_2$, we introduce a fresh element error as a dummy error state to be verified, and add the reduction from the original error states to error: $\langle\, \mathsf{crit}_0, \mathsf{crit}_1, \sigma \,\rangle \to_{\mathcal{A}_2} \mathsf{error}$, where $\sigma \in \mathcal{S}$. We

$$\dfrac{\dfrac{\dfrac{\dfrac{(2)\dagger \quad (3)\ddagger}{(8)}\ \dfrac{\dfrac{(3)\ddagger}{(9)\lozenge}\,(\textsc{Der})}{(4)}\,(\textsc{Der})\qquad \dfrac{(9)\lozenge}{(5)}\,(\textsc{Der})}{(2)\dagger}\,(\textsc{Der})\qquad \dfrac{\dfrac{\dfrac{(2)\dagger}{(10)\blacklozenge}\,(\textsc{Der})}{(6)}\,(\textsc{Der})\qquad \dfrac{(10)\blacklozenge \quad \dfrac{(2)\dagger \quad (3)\ddagger}{(11)}\,(\textsc{Der})}{(7)}\,(\textsc{Der})}{(3)\ddagger}\,(\textsc{Der})}{(1)\ \{\langle \mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, m\rangle \mid m \in \{0,1\}\} \Rightarrow \emptyset}\,(\textsc{Der})$$

(2) $\{\langle \mathsf{wait}_0, \mathsf{noncrit}_1, \mathsf{true}, \mathsf{false}, 1\rangle\} \Rightarrow \emptyset$
(3) $\{\langle \mathsf{noncrit}_0, \mathsf{wait}_1, \mathsf{false}, \mathsf{true}, 0\rangle\} \Rightarrow \emptyset$
(4) $\{\langle \mathsf{crit}_0, \mathsf{noncrit}_1, \mathsf{true}, \mathsf{false}, 1\rangle\} \Rightarrow \emptyset$
(5) $\{\langle \mathsf{wait}_0, \mathsf{wait}_1, \mathsf{true}, \mathsf{true}, 0\rangle\} \Rightarrow \emptyset$
(6) $\{\langle \mathsf{wait}_0, \mathsf{wait}_1, \mathsf{true}, \mathsf{true}, 1\rangle\} \Rightarrow \emptyset$
(7) $\{\langle \mathsf{noncrit}_0, \mathsf{crit}_1, \mathsf{false}, \mathsf{true}, 0\rangle\} \Rightarrow \emptyset$
(8) $\{\langle \mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, 1\rangle\} \Rightarrow \emptyset$
(9) $\{\langle \mathsf{crit}_0, \mathsf{wait}_1, \mathsf{true}, \mathsf{true}, 0\rangle\} \Rightarrow \emptyset$
(10) $\{\langle \mathsf{wait}_0, \mathsf{crit}_1, \mathsf{true}, \mathsf{true}, 1\rangle\} \Rightarrow \emptyset$
(11) $\{\langle \mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, 0\rangle\} \Rightarrow \emptyset$

Figure 7 An APR proof for $\{\langle \mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, m\rangle \mid m \in \{0,1\}\} \Rightarrow \emptyset$, where †, ‡, ◊, and ⧫ indicate bud-companion relationships.

let $\mathcal{A}_2'$ be the modified ARS $(\{\langle p_0, p_1, \sigma\rangle \mid p_0 \in \mathit{State}_0,\ p_1 \in \mathit{State}_1,\ \sigma \in \mathcal{S}\}, \to_{\mathcal{A}_2'})$, where $\to_{\mathcal{A}_2'} = \to_{\mathcal{A}_2} \cup \{(\langle \mathsf{crit}_0, \mathsf{crit}_1, \sigma\rangle, \mathsf{error}) \mid \sigma \in \mathcal{S}\}$. The race freedom is reduced to the APR predicate (1) $\{\langle \mathsf{noncrit}_0, \mathsf{noncrit}_1, \mathsf{false}, \mathsf{false}, m\rangle \mid m \in \{0,1\}\} \Rightarrow \emptyset$. We obtain an APR proof illustrated in Figure 7, and thus, by Theorem 4.9, the APR predicate (1) is partially valid w.r.t. $\mathcal{A}_2'$. Therefore, by Theorem 5.2, the AITS defined by the program graphs in Figure 6 is race-free and, by Proposition 2.4 (5), there is no normal form reachable from an initial state.

For $P \Rightarrow Q$ being a safety APR predicate for $E \subseteq \mathit{NF}_{\mathcal{A}}$, the target set Q is required to satisfy that $\{t \in \mathit{NF}_{\mathcal{A}} \setminus E \mid \exists s \in P.\ s \to_{\mathcal{A}}^* t\} \subseteq Q$, while it suffices to satisfy $\{t \in \mathit{NF}_{\mathcal{A}} \setminus E \mid \exists s \in P.\ s \to_{\mathcal{A}}^* t\} = Q$. It may be difficult for a given rewrite system $\mathcal{A}$ to compute the set $\{t \in \mathit{NF}_{\mathcal{A}} \setminus E \mid \exists s \in P.\ s \to_{\mathcal{A}}^* t\}$, and thus we allow to use an over-approximation Q such that $\{t \in \mathit{NF}_{\mathcal{A}} \setminus E \mid \exists s \in P.\ s \to_{\mathcal{A}}^* t\} \subseteq Q$. On the other hand, as in [15], a simpler modification is useful: We introduce a fresh element any to A and extend $\to_{\mathcal{A}}$ to $\to_{\mathcal{A}} \cup \{(s, \mathsf{any}) \mid s \in A \setminus E\}$. This modification is not an approximation.

▶ **Theorem 5.4.** *Let $P \Rightarrow Q$ be a safety APR predicate for $E \subseteq \mathit{NF}_{\mathcal{A}}$. Then, $\mathcal{A} \models^{\forall}_{\mathrm{partial}} P \Rightarrow Q$ if and only if $(A \cup \{\mathsf{any}\}, \to_{\mathcal{A}} \cup \{(s, \mathsf{any}) \mid s \in A \setminus E\}) \models^{\forall}_{\mathrm{partial}} P \Rightarrow \{\mathsf{any}\}$.*

One may think that non-reachability analysis is more suitable for safety properties. However, APR analysis is essentially very similar to non-reachability analysis:

- For non-reachability from P to E, all states reachable from P are examined, unless an error state reachable from P is detected.
- During the construction of an APR pre-proof, all states reachable from P are examined, unless a subset P' with $P' \cap E \neq \emptyset$ is detected.

6 Reduction of Liveness Properties to APR Problems

Informally speaking, *liveness properties* state that "something good" will eventually happen for every execution [2, Section 3.4]. As a liveness property w.r.t. a system, a source set P, and a set Q for "something good", we consider the property that any (possibly non-terminating) execution of the system starting from any state in P reaches a state in Q. This kind of safety properties is formulated as a problem for ARSs as follows:

Instance An ARS $\mathcal{A} = (A, \to_{\mathcal{A}})$ and sets P, Q $(\subseteq A)$
Question Does *every* (possibly infinite) execution path $s_0 \to_{\mathcal{A}} s_1 \to_{\mathcal{A}} \cdots$ with $s_0 \in P$ include an element in Q (i.e., $s_i \in Q$ for some $i \geq 0$)?

Unfortunately, this problem cannot be reduced to partial validity of $P \Rightarrow Q$ w.r.t. $\mathcal{A}$, while $P \Rightarrow Q$ seems natural for liveness properties. The shortcoming is that partial validity does not take into account any infinite execution path. In this section, we introduce a stronger validity, called *total validity*, which takes into account *all* (possibly infinite) execution paths. Then, for a partially valid APR predicate with an APR proof, we show a necessary and sufficient condition for the tree to ensure total validity, showing how to apply APR analysis to verification of liveness properties.

6.1 Total Validity of APR Predicates w.r.t. ARSs

We first introduce a stronger validity which takes into account all execution paths. Total validity of $P \Rightarrow Q$ w.r.t. $\mathcal{A}$ ensures that every execution path starting from an element in P eventually reaches an element in Q.

▶ **Definition 6.1** (total validity). *An APR predicate $P \Rightarrow Q$ over A is said to be* totally valid *w.r.t. $\mathcal{A}$, written as $\mathcal{A} \models^{\forall}_{\text{total}} P \Rightarrow Q$, if every (possibly infinite) execution path $s_0 \to_{\mathcal{A}} s_1 \to_{\mathcal{A}} \cdots$ with $s_0 \in P$ includes an element in Q, i.e., $s_i \in Q$ for some $i \geq 0$. We write $\mathcal{A} \not\models^{\forall}_{\text{total}} P \Rightarrow Q$ if $P \Rightarrow Q$ is not totally valid w.r.t. $\mathcal{A}$.*

▶ **Example 6.2.** Let us continue with Example 5.3. Starvation freedom for the process P_0 specified by PG_0 in Figure 6 – P_0 can reach crit_0 from wait_0 – is reduced to total validity of the following APR predicate:

$$(12)\ \{\langle\, \mathsf{wait}_0, p_1, \sigma \,\rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}, \sigma(b_0) = \mathsf{true}\} \Rightarrow \{\langle\, \mathsf{crit}_0, p_1, \sigma \,\rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}\}$$

By definition, it is clear that total validity implies partial one. In other words, partial validity is a necessary condition for total one. Thus, to prove total validity, we should first prove partial validity and we usually attempt to construct an APR proof. However, the existence of such an APR proof does not ensure total validity. For this reason, we need an additional condition. To this end, in the next section, we will show a sufficient condition for APR proofs to additionally imply total validity.

6.2 A Criterion for Total Validity of Partial Valid APR Predicates

Let us consider an APR proof and its proof graph for $P \Rightarrow Q$. If the proof graph has no cycle, then there is no infinite execution path that does not include any element in Q; if there exists such a path, then either it is impossible to obtain a finite APR proof or the proof graph has a cycle. Recall that APR proofs are finite derivation trees.

▶ **Theorem 6.3.** *Let $((V, \mathrm{a}, \mathrm{r}, c), \xi)$ be a partially valid APR proof for an APR predicate $P \Rightarrow Q$ w.r.t. $\mathcal{A}$, and (V', E) be the proof graph of the APR proof. Then, (V', E) is acyclic if and only if $\mathcal{A} \models^{\forall}_{\text{total}} P \Rightarrow Q$.*

Once we construct an APR proof, the total validity is decidable.

▶ **Corollary 6.4.** *Let $((V, \mathrm{a}, \mathrm{r}, c), \xi)$ be an APR proof for a partially valid APR predicate $P \Rightarrow Q$ w.r.t. $\mathcal{A}$. Then, total validity of $P \Rightarrow Q$ w.r.t. $\mathcal{A}$ is decidable.*

Proof. It is decidable whether a finite graph is acyclic, and thus, by Theorem 6.3, totally validity is decidable. ◀

$$\dfrac{\dfrac{\dfrac{}{(17)}\,(\textsc{Axiom})}{(13)}\,(\textsc{Subs}) \quad \dfrac{\dfrac{\dfrac{}{(17)}\,(\textsc{Axiom})}{(18)}\,(\textsc{Subs})}{(14)\dagger}\,(\textsc{Der}) \quad \dfrac{\dfrac{\dfrac{}{(17)}\,(\textsc{Axiom})}{(21)}\,(\textsc{Subs}) \qquad (14)\dagger}{(19)}\,(\textsc{Der}) \quad \dfrac{\dfrac{}{(17)}\,(\textsc{Axiom})}{(20)}\,(\textsc{Subs}) \qquad (14)\dagger}{(15)}\,(\textsc{Der}) \qquad (16)}{(12)\ \{\langle \mathsf{wait}_0, p_1, \sigma \rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}, \sigma(b_0) = \mathsf{true}\} \Rightarrow \{\langle \mathsf{crit}_0, p_1, \sigma \rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}\}}\,(\textsc{Der})$$

(13) $\{\langle \mathsf{crit}_0, p_1, \mathsf{true}, v_1, m \rangle \mid v_1 \in \mathbb{B}, m \in \{0,1\}, m = 0 \vee v_1 = \mathsf{false}\} \Rightarrow \{\langle \mathsf{crit}_0, p_1, \sigma \rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}\}$
(14) $\{\langle \mathsf{wait}_0, \mathsf{wait}_1, \mathsf{true}, \mathsf{true}, 0 \rangle\} \Rightarrow \{\langle \mathsf{crit}_0, p_1, \sigma \rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}\}$
(15) $\{\langle \mathsf{wait}_0, \mathsf{crit}_1, \mathsf{true}, v_1, 1 \rangle \mid v_1 \in \mathbb{B}\} \Rightarrow \{\langle \mathsf{crit}_0, p_1, \sigma \rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}\}$
(16) $\{\langle \mathsf{wait}_0, \mathsf{noncrit}_1, \mathsf{true}, \mathsf{false}, m \rangle \mid m \in \{0,1\}\} \Rightarrow \{\langle \mathsf{crit}_0, p_1, \sigma \rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}\}$
(17) $\emptyset \Rightarrow \{\langle \mathsf{crit}_0, p_1, \sigma \rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}\}$
(18) $\{\langle \mathsf{crit}_0, \mathsf{wait}_1, \mathsf{true}, \mathsf{true}, 0 \rangle\} \Rightarrow \{\langle \mathsf{crit}_0, p_1, \sigma \rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}\}$
(19) $\{\langle \mathsf{wait}_0, \mathsf{noncrit}_1, \mathsf{true}, \mathsf{false}, 1 \rangle\} \Rightarrow \{\langle \mathsf{crit}_0, p_1, \sigma \rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}\}$
(20) $\{\langle \mathsf{crit}_0, \mathsf{noncrit}_1, \mathsf{true}, \mathsf{false}, m \rangle \mid m \in \{0,1\}\} \Rightarrow \{\langle \mathsf{crit}_0, p_1, \sigma \rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}\}$
(21) $\{\langle \mathsf{crit}_0, \mathsf{noncrit}_1, \mathsf{true}, \mathsf{false}, 1 \rangle\} \Rightarrow \{\langle \mathsf{crit}_0, p_1, \sigma \rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}\}$
(22) $\{\langle \mathsf{wait}_0, \mathsf{wait}_1, \mathsf{true}, \mathsf{true}, 0 \rangle\} \Rightarrow \{\langle \mathsf{crit}_0, p_1, \sigma \rangle \mid p_1 \in \mathit{State}_1, \sigma \in \mathcal{S}\}$

Figure 8 An APR proof for (12), where † indicates bud-companion relationships.

We call a closed APR pre-proof *acyclic* if its proof graph is acyclic. Notice that an acyclic APR pre-proof may have a bud node and its companion. APR pre-proofs are simpler variants of cyclic proofs and the bud-companion relationship leads to the terminology "cyclic". On the other hand, the existence of bud nodes and companions in an APR pre-proof does not always induce an actual circularity: Regarding an APR pre-proof for an APR predicate, if its proof graph is acyclic, then the APR pre-proof can be expanded to another APR pre-proof for the APR predicate, which does not include any bud node and its companion (cf. [3, Section 5]). For such an APR pre-proof, the bud-companion relationship enables us to reduce the search space and thus the size of constructing trees.

As a consequence of Theorem 6.3, for an APR predicate $P \Rightarrow Q$, the existence of an acyclic APR pre-proof is a sufficient condition for total validity of $P \Rightarrow Q$ w.r.t. $\mathcal{A}$.

▶ **Corollary 6.5.** *For an APR predicate $P \Rightarrow Q$ over A, if there exists an acyclic APR proof for $P \Rightarrow Q$ w.r.t. $\mathcal{A}$, then $\mathcal{A} \models^{\forall}_{\mathrm{total}} P \Rightarrow Q$.*

▶ **Example 6.6.** Let us continue with Example 6.2. We have an APR proof for the APR predicate (12) shown in Figure 8, and the APR proof is acyclic because its proof graph shown in Figure 9 is acyclic. Therefore, by Corollary 6.5, the APR predicate (12) is totally valid w.r.t. $\mathcal{A}_2$, and thus the AITS is starvation free for P_0.

7 Related Work

The work in this paper is the adaptation of the APR frameworks in [7, 15] to ARSs, providing an abstract foundation for APR analysis. The cyclic-proof system for partial validity of APR predicates in this paper is a simplified adaptation of the well-known cyclic-proof system CLKID$^\omega$ [4, Chapter 5]. Compared with sequents, APR predicates are very simple, and our APR proofs always satisfy the global trace condition.

Runtime-error verification by means of APR analysis w.r.t. constrained rewrite systems such as LCTRSs has been investigated. $\mathbb{K}$ framework [25] is a more general setting of constrained rewriting than LCTRSs, and the proof system for partial validity of APR

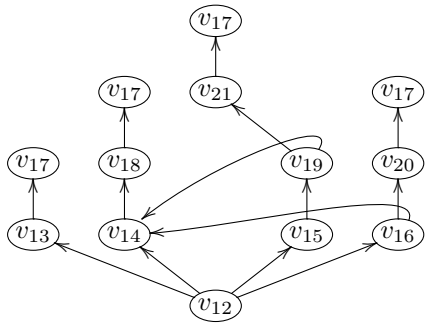


Figure 9 The proof graph of the APR proof for (12) in Figure 8, where node $a(v_i) = (i)$.

predicates consisting of constrained terms has been implemented in $\mathbb{K}$ [8, 9]. The race-freedom of Peterson's mutual exclusion algorithm has been proved in [9] by means of the APR approach, while starvation freedom is not considered. A comparison of *all-path reachability logic* with CTL* can be seen in [9].

Our previous APR-based approach in [15] to safety verification is to add rules to a given LCTRS so as to make any finite prefix of all (possible infinite) execution paths finite execution paths of the modified LCTRS. On the other hand, as described in Section 5, it suffices to include in the target set all irreducible states that are not error states, and we formulated the necessary condition as safety APR predicates for sets of error states. However, for the inclusion of all non-error irreducible states in the target set, it is enough to add the reduction from all states to a freshly introduced dummy state that is irreducible w.r.t. the modified ARS. Viewed in this light, in practice, these two approaches are similar, and the approach in this paper is an abstract framework of the previous one for LCTRSs.

Our previous work [14, 16] for the verification of starvation freedom by means of APR analysis with LCTRSs reduces starvation freedom to safety APR predicates, and does not use the approach to liveness properties in this paper. On the other hand, the previous work [16] deals with *process-fairness* for starvation freedom.

8 Conclusion

In this paper, we first reformulated inference rules for partial validity of APR predicates w.r.t. ARSs, and adapted the cyclic-proof system for partial validity w.r.t. LCTRSs to ARSs. The reformulated framework includes a disproof rule. Then, we showed how to apply APR analysis to safety verification. Finally, we introduced total validity of APR predicates w.r.t. ARSs and showed that if there is an acyclic APR proof for an APR predicate w.r.t. an ARS, then the APR predicate is totally valid w.r.t. the ARS. Total validity of APR predicates can be used for liveness verification.

Our APR pre-proofs do not consider the generalization for bud nodes. To be more precise, in the current definition, a bud node and its companion have the same APR predicate. As described in Footnote 2, the relaxation of the bud-companion relationship – an APR predicate can be more general than that of its companion – is one of our future directions for the practical use of APR analysis for runtime-error verification. The introduction of *process-fairness* to the abstract APR framework formulated in this paper is also a future direction to make APR-based verification more practical. Another future direction of this research is to deal with the full version of the rule Circ, i.e., "cut" in APR pre-proofs.

References

1 Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998. doi:10.1145/505863.505888.

2 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

3 James Brotherston. Cyclic proofs for first-order logic with inductive definitions. In Bernhard Beckert, editor, *Proceedings of the 14th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, volume 3702 of *Lecture Notes in Computer Science*, pages 78–92. Springer, 2005. doi:10.1007/11554554_8.

4 James Brotherston. *Sequent Calculus Proof Systems for Inductive Definitions*. PhD thesis, University of Edinburgh, November 2006.

5 Ştefan Ciobâcă and Dorel Lucanu. A coinductive approach to proving reachability properties in logically constrained term rewriting systems. *CoRR*, abs/1804.08308, 2018. doi:10.48550/arXiv.1804.08308.

6 Ştefan Ciobâcă, Dorel Lucanu, and Andrei-Sebastian Buruiană. Operationally-based program equivalence proofs using LCTRSs. *Journal of Logical and Algebraic Methods in Programming*, 135:1–22, 2023. doi:10.1016/j.jlamp.2023.100894.

7 Ştefan Ciobâcă and Dorel Lucanu. A coinductive approach to proving reachability properties in logically constrained term rewriting systems. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Proceedings of the 9th International Joint Conference on Automated Reasoning*, volume 10900 of *Lecture Notes in Computer Science*, pages 295–311. Springer, 2018. doi:10.1007/978-3-319-94205-6_20.

8 Andrei Ştefănescu, Ştefan Ciobâcă, Radu Mereuta, Brandon M. Moore, Traian-Florin Şerbănuţă, and Grigore Roşu. All-path reachability logic. In Gilles Dowek, editor, *Proceedings of the Joint International Conference on Rewriting and Typed Lambda Calculi*, volume 8560 of *Lecture Notes in Computer Science*, pages 425–440. Springer, 2014. doi:10.1007/978-3-319-08918-8_29.

9 Andrei Ştefănescu, Ştefan Ciobâcă, Radu Mereuta, Brandon M. Moore, Traian-Florin Şerbănuţă, and Grigore Roşu. All-path reachability logic. *Logical Methods in Computer Science*, 15(2), 2019. doi:10.23638/LMCS-15(2:5)2019.

10 Carsten Fuhs, Cynthia Kop, and Naoki Nishida. Verifying procedural programs via constrained rewriting induction. *ACM Transactions on Computational Logic*, 18(2):14:1–14:50, 2017. doi:10.1145/3060143.

11 Yoshiaki Kanazawa and Naoki Nishida. On transforming functions accessing global variables into logically constrained term rewriting systems. In Joachim Niehren and David Sabel, editors, *Proceedings of the 5th International Workshop on Rewriting Techniques for Program Transformations and Evaluation*, volume 289 of *Electronic Proceedings in Theoretical Computer Science*, pages 34–52. Open Publishing Association, 2019. doi:10.4204/EPTCS.289.3.

12 Yoshiaki Kanazawa, Naoki Nishida, and Masahiko Sakai. On representation of structures and unions in logically constrained rewriting. IEICE Technical Report SS2018-38, the Institute of Electronics, Information and Communication Engineers, 2019. Vol. 118, No. 385, pp. 67–72, in Japanese.

13 Misaki Kojima. *Runtime-Error Verification by Reduction to All-Path Reachability Problems of Constrained Rewriting*. PhD thesis, Graduate School of Informatics, Nagoya University, 2024. URL: https://nagoya.repo.nii.ac.jp/records/2010041.

14 Misaki Kojima and Naoki Nishida. From starvation freedom to all-path reachability problems in constrained rewriting. In Michael Hanus and Daniela Inclezan, editors, *Proceedings of the 25th International Symposium on Practical Aspects of Declarative Languages*, volume 13880 of *Lecture Notes in Computer Science*, pages 161–179. Springer Nature Switzerland, 2023. doi:10.1007/978-3-031-24841-2_11.

15 Misaki Kojima and Naoki Nishida. Reducing non-occurrence of specified runtime errors to all-path reachability problems of constrained rewriting. *Journal of Logical and Algebraic Methods in Programming*, 135:1–19, 2023. doi:10.1016/j.jlamp.2023.100903.

16 Misaki Kojima and Naoki Nishida. On solving all-path reachability problems for starvation freedom of concurrent rewrite systems under process fairness. In Laura Kovács and Ana Sokolova, editors, *Proceedings of the 18th International Conference on Reachability Problems*, volume 15050 of *Lecture Notes in Computer Science*, pages 54–70. Springer, 2024. doi:10.1007/978-3-031-72621-7_5.

17 Misaki Kojima and Naoki Nishida. A sufficient condition of logically constrained term rewrite systems for decidability of all-path reachability problems with constant destinations. *Journal of Information Processing*, 32:417–435, 2024. doi:10.2197/IPSJJIP.32.417.

18 Misaki Kojima and Naoki Nishida. Abstract Framework for All-Path Reachability Analysis toward Safety and Liveness Verification (Full Version). *CoRR*, abs/2602.04641, 2026. doi:10.48550/arXiv.2602.04641.

19 Misaki Kojima, Naoki Nishida, and Yutaka Matsubara. Transforming concurrent programs with semaphores into logically constrained term rewrite systems. *Journal of Logical and Algebraic Methods in Programming*, 143:1–23, 2025. doi:10.1016/j.jlamp.2024.101033.

20 Cynthia Kop and Naoki Nishida. Term rewriting with logical constraints. In Pascal Fontaine, Christophe Ringeissen, and Renate A. Schmidt, editors, *Proceedings of the 9th International Symposium on Frontiers of Combining Systems*, volume 8152 of *Lecture Notes in Computer Science*, pages 343–358. Springer, 2013. doi:10.1007/978-3-642-40885-4_24.

21 Ayuka Matsumi, Naoki Nishida, Misaki Kojima, and Donghoon Shin. On singleton self-loop removal for termination of LCTRSs with bit-vector arithmetic. In Akihisa Yamada, editor, *Proceedings of the 19th International Workshop on Termination*, pages 1–6, August 2023. doi:10.48550/arXiv.2307.14094.

22 Dragana Milovančević, Carsten Fuhs, and Viktor Kunčak. Proving termination of scala programs by constrained term rewriting. In Cynthia Kop and Janis Voigtländer, editors, *Informal Proceedings of the 11th International Workshop on Rewriting Techniques for Program Transformations and Evaluation*, pages 13–25, July 2025. URL: https://wpte2025.github.io/pre-proceedings.pdf#page=15.

23 Naoki Nishida and Sarah Winkler. Loop detection by logically constrained term rewriting. In Ruzica Piskac and Philipp Rümmer, editors, *Proceedings of the 10th Working Conference on Verified Software: Theories, Tools, and Experiments*, volume 11294 of *Lecture Notes in Computer Science*, pages 309–321. Springer, 2018. doi:10.1007/978-3-030-03592-1_18.

24 Enno Ohlebusch. *Advanced Topics in Term Rewriting*. Springer, 2002. doi:10.1007/978-1-4757-3661-8.

25 Grigore Roşu and Traian-Florin Şerbănuţă. An overview of the K semantic framework. *Journal of Logic and Algebraic Programming*, 79(6):397–434, 2010. doi:10.1016/j.jlap.2010.03.012.

26 Sarah Winkler and Aart Middeldorp. Completion for logically constrained rewriting. In Hélène Kirchner, editor, *Proceedings of the 3rd International Conference on Formal Structures for Computation and Deduction*, volume 108 of *Leibniz International Proceedings in Informatics*, pages 30:1–30:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSCD.2018.30.

A Omitted Proofs

▶ **Theorem 3.3.** *For any ARS $\mathcal{A}$, $\nu\widehat{\mathsf{DVP}} = \nu\widehat{\mathsf{DVP}_+}$.*

Proof. By definition, we have that $\nu\widehat{\mathsf{DVP}} \subseteq \nu\widehat{\mathsf{DVP}_+}$, because the applications of Subsumption and Step in DVP can be simulated by Subs and either Axiom or Der in DVP_+. We show that $\nu\widehat{\mathsf{DVP}} \supseteq \nu\widehat{\mathsf{DVP}_+}$. The application of Axiom in DVP_+ can be simulated by Subsumption in DVP. The application of Subs in DVP_+ is followed by either Axiom or Der in DVP_+. Let us consider the following case:

$$\frac{P_1 \Rightarrow Q \qquad \dots \qquad P_n \Rightarrow Q}{P \Rightarrow Q}\ (\textsc{Subs})$$

where $P \cap Q \neq \emptyset$, $P \setminus Q = P_1 \cup \cdots \cup P_n$ for some $n > 0$, and if $n > 1$ then $P_i \neq \emptyset$ for all $1 \leq i \leq n$. We make a case analysis depending on whether $P \setminus Q = \emptyset$ or not.

- Case where $P \setminus Q = \emptyset$. By definition, we have that $P \subseteq Q$. Therefore, the successive application of Subs and Axiom can be simulated by Subsumption in DVP.
- Case where $P \setminus Q \neq \emptyset$. By definition, we have that $P_i \neq \emptyset$ for all $1 \leq i \leq n$. Since $P_i \subseteq P \setminus Q$, we have that $P_i \cap Q = \emptyset$. Since $(P \Rightarrow Q) \in \nu\widehat{\mathsf{DVP}_+}$, at least one rule is applicable to $P_i \Rightarrow Q$ for each $i \in \{1, \ldots, n\}$. By Proposition 3.2, the applicable rule is Der only. Thus, for each $i \in \{1, \ldots, n\}$, we have the following subtree:

$$\frac{P_{i,1} \Rightarrow Q \qquad \cdots \qquad P_{i,m_i} \Rightarrow Q}{P_i \Rightarrow Q}\ (\textsc{Der})$$

  where $\partial_{\mathcal{A}}(P_i) = P_{i,1} \cup \cdots \cup P_{i,m_i}$ for some $m_i > 0$, and $P_{i,j} \neq \emptyset$ for all $1 \leq j \leq m_i$. By definition, we have that $P \setminus Q = \bigcup_{i=1}^{n} P_i$ and $P_i \setminus Q = P_i$ for all $1 \leq i \leq n$. Hence, we have that $\partial_{\mathcal{A}}(P_i \setminus Q) = \partial_{\mathcal{A}}(P_i) = \bigcup_{j=1}^{m_i} P_{i,j}$, and thus $\partial_{\mathcal{A}}(P \setminus Q) = \partial_{\mathcal{A}}(\bigcup_{i=1}^{n} P_i) = \bigcup_{i=1}^{n}\bigcup_{j=1}^{m_i} P_{i,j}$. Thus, we have the following tree of DVP:

$$\frac{(\bigcup_{i=1}^{n}\bigcup_{j=1}^{m_i} P_{i,j}) \Rightarrow Q}{P \Rightarrow Q}\ (\textsc{Step})$$

  Since $(P_i \Rightarrow Q) \in \nu\widehat{\mathsf{DVP}_+}$, we have that $(P_{i,j} \Rightarrow Q) \in \nu\widehat{\mathsf{DVP}_+}$ for all $1 \leq i \leq n$ and $1 \leq j \leq m_i$. By induction, we have that for all $1 \leq i \leq n$, $(P_{i,1} \Rightarrow Q), \ldots, (P_{i,m_i} \Rightarrow Q) \in \nu\widehat{\mathsf{DVP}}$, and thus $\mathcal{A} \models^{\forall}_{\text{partial}} P_{i,j} \Rightarrow Q$ for all $1 \leq i \leq n$ and $1 \leq j \leq m_i$. It follows from Proposition 2.4 that $\mathcal{A} \models^{\forall}_{\text{partial}} (\bigcup_{i=1}^{n}\bigcup_{j=1}^{m_i} P_{i,j}) \Rightarrow Q$ and thus $((\bigcup_{i=1}^{n}\bigcup_{j=1}^{m_i} P_{i,j}) \Rightarrow Q) \in \nu\widehat{\mathsf{DVP}}$. Therefore, we have that $(P \Rightarrow Q) \in \nu\widehat{\mathsf{DVP}}$.

The application of Der in DVP_+ to $(P \Rightarrow Q)$ is a special case of the successive application of Subs and Der in DVP_+, and thus $(P \Rightarrow Q) \in \nu\widehat{\mathsf{DVP}}$. ◀

▶ **Theorem 4.9.** *Let $P \Rightarrow Q$ be an APR predicate over A. Then, both of the following statements hold:*

(1) *if there exists an APR proof for $P \Rightarrow Q$ w.r.t. $\mathcal{A}$, then $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$, and*

(2) *if there exists an APR disproof of $P \Rightarrow Q$ w.r.t. $\mathcal{A}$, then $\mathcal{A} \not\models^{\forall}_{\text{partial}} P \Rightarrow Q$.*

Proof. We first prove the first claim, which can be proved analogously to the proof in [7, Theorem 3]. We proceed by contradiction. Assume that there exists an APR proof of $P \Rightarrow Q$ and $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$. Let $((V, \mathsf{a}, r, c), \xi)$ be an APR proof for $P \Rightarrow Q$ with root node v_0, and (V', E') be the proof graph of $((V, \mathsf{a}, r, c), \xi)$ such that $v_0 \in V'$, where $\mathsf{a}(v_0) = (P \Rightarrow Q)$. Since $\mathcal{A} \not\models^{\forall}_{\text{partial}} P \Rightarrow Q$, there exists an execution path $s_0 \to_{\mathcal{A}} s_1 \to_{\mathcal{A}} \cdots \to_{\mathcal{A}} v_k$ such that $s_0 \in P$, $\{s_0, s_1, \ldots, s_k\} \cap Q = \emptyset$, and $s_k \in NF_{\mathcal{A}}$. Since $\mathsf{a}(v_0) = (P \Rightarrow Q)$, we have that $s_0 \in a_L(v_0)$. It follows from Proposition 4.6 that there exist nodes $v_1, \ldots, v_k \in V'$ such that $v_0\ (\to_{\textsc{Der}} \cup \to_{\textsc{Subs}})\ v_1\ (\to_{\textsc{Der}} \cup \to_{\textsc{Subs}}) \cdots (\to_{\textsc{Der}} \cup \to_{\textsc{Subs}})\ v_k$. Since $s_k \in NF_{\mathcal{A}}$, by the construction of proof graphs, v_k is a node in V such that $r(v_k) = \textsc{Dis}$, and thus $((V, \mathsf{a}, r, c), \xi)$ is an APR disproof. This contradicts the assumption that $((V, \mathsf{a}, r, c), \xi)$ is an APR proof.

Next, we prove the second claim. Let $((V, \mathsf{a}, r, c), \xi)$ be an APR disproof for $P \Rightarrow Q$ with root node v_0, where $\mathsf{a}(v_0) = (P \Rightarrow Q)$. Then, there exists node $v, v' \in V$ such that $r(v) = \textsc{Dis}$, $c(v) = v'$, and $\mathsf{a}(v') = \bot$. Let $\mathsf{a}(v) = (P' \Rightarrow Q)$. Then, by the condition of v, v', we have that $P' \neq \emptyset$, $P' \cap Q = \emptyset$, and $P' \cap NF_{\mathcal{A}} \neq \emptyset$. Thus, there exists an element $s \in P'$ such that $s \in NF_{\mathcal{A}}$ and $s \notin Q$. By the construction of APR pre-proofs, there exist nodes $v_1, \ldots, v_k$ such that $v_0\ (\to_{\textsc{Der}} \cup \to_{\textsc{Subs}})\ v_1\ (\to_{\textsc{Der}} \cup \to_{\textsc{Subs}}) \cdots (\to_{\textsc{Der}} \cup \to_{\textsc{Subs}})\ v_k\ (\to_{\textsc{Der}} \cup \to_{\textsc{Subs}})\ v\ (\to_{\textsc{Der}} \cup \to_{\textsc{Subs}})\ v'$. It is clear that Proposition 4.6 holds for paths of derivation trees, and thus, by Proposition 4.6, there exist elements $s_0, s_1, \ldots, s_k$ such

that $s_0 \to_{\mathcal{A}}^{=} s_1 \to_{\mathcal{A}}^{=} \cdots \to_{\mathcal{A}}^{=} s_k \to_{\mathcal{A}}^{=} s$ and $s_i \in a_L(v_i)$ and $a_L(v_i) \cap Q = \emptyset$ for all $0 \leq i \leq k$. Thus, we have that $s_i \notin Q$ for all $0 \leq i \leq k$, and thus $s_0 \to_{\mathcal{A}}^{=} s_1 \to_{\mathcal{A}}^{=} \cdots \to_{\mathcal{A}}^{=} s_k \to_{\mathcal{A}}^{=} s$ is a finite execution path that does not include any element in Q. Therefore, we have that $\mathcal{A} \not\models^{\forall}_{\text{partial}} P \Rightarrow Q$. ◀

▶ **Theorem 4.10.** *Let $P \Rightarrow Q$ be an APR predicate over A, and $\partial^*_{\mathcal{A}}(P) = \{t \in A \mid \exists s \in P.\ s \to^*_{\mathcal{A}} t\}$. Then, both of the following statements hold:*

(1) *If $\partial^*_{\mathcal{A}}(P)$ is finite, then there exists either an APR proof or disproof for $P \Rightarrow Q$, and*

(2) *if $\partial^*_{\mathcal{A}}(P)$ is finite and the emptiness, intersection emptiness, and $\mathcal{A}$-runnability problems for A are decidable, then partial validity of $P \Rightarrow Q$ w.r.t. $\mathcal{A}$ is decidable.*

Proof. We first show a naive construction of APR pre-proofs of $P \Rightarrow Q$, where we do not split any source sets. It follows from Proposition 3.6 that for each APR predicate, there is exactly one applicable rule in $\mathsf{DVP}_\perp$. Let v_0 be the root of a derivation tree we construct now, where $a(v_0) = (P \Rightarrow Q)$. In the following, we use V as a set of nodes that have already been applied a rule in $\mathsf{DVP}_\perp$ or considered a bud node; we use V_{Der} as a set of nodes that can be companions of other nodes; we use U as a queue of nodes that have not been applied any rule in $\mathsf{DVP}_\perp$ yet. Then, we construct an APR pre-proof by breadth-first search (cf. [17]):

1. $V := \{v_0\}$, $V_{\text{Der}} := \emptyset$, and insert v_0 to U.
2. Repeat the following until U has no node, and then return $((V, a, r, c), \xi)$:
   a. Take a node v from U, and $V := V \cup \{v\}$.
   b. If there exists a node $v' \in V_{\text{Der}}$ with $a(v') = a(v)$, then $\xi(v) := v'$ and skip **c**.
   c. If Axiom is applicable to $a(v)$, then $r(v) :=$ Axiom, $c(v) := \epsilon$; if Subs is applicable to $a(v)$, then create a node v', insert v' to U, $r(v) :=$ Subs, $a(v') := ((a_L(v) \setminus a_R(v)) \Rightarrow a_R(v))$, and $c(v) := v'$; if Der is applicable to $a(v)$, then create a node v', insert v' to U, $r(v) :=$ Der, $a(v') := (\partial_{\mathcal{A}}(a_L(v)) \Rightarrow a_R(v))$, $c(v) := v'$, and $V_{\text{Der}} := V_{\text{Der}} \cup \{v\}$; if Dis is applicable to $a(v)$, then create a node v', $V := V \cup \{v'\}$, $r(v) :=$ Dis, $a(v') := \bot$, and $c(v) := v'$.

It is clear that the above procedure is deterministic and the output is a closed APR pre-proof.

To prove partial validity of $P \Rightarrow Q$, it suffices to consider APR predicates $P' \Rightarrow Q$ such that $P' \subseteq \partial^*_{\mathcal{A}}(P)$. Since $\partial^*_{\mathcal{A}}(P)$ is finite, there are only finitely many APR predicates to be considered. If the above procedure does not halt, then it is clear that there are infinitely many APR predicates to be considered for partial validity of $P \Rightarrow Q$, and this contradicts the finiteness of APR predicates to be considered. If there exists a node v with $a(v) = \bot$, then the output is an APR disproof, and otherwise, it is an APR proof. Therefore, the first claim holds. If the side conditions of rules in $\mathsf{DVP}_\perp$ are decidable, then the procedure above halts for any APR predicate if $\partial^*_{\mathcal{A}}(P)$ is finite. Therefore, the second claim holds. ◀

▶ **Theorem 5.2.** *Let $P, Q \subseteq A$, $E \subseteq NF_{\mathcal{A}}$, and $P \Rightarrow Q$ be a safety APR predicate for E w.r.t. $\mathcal{A}$. Then, $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$ if and only if there is no finite execution path of $\mathcal{A}$ that starts with an element in P and includes an element in E.*

Proof. We prove the *only-if* part by contradiction. Assume that $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$ and there is a finite execution path $s_0 \to_{\mathcal{A}} s_1 \to_{\mathcal{A}} \cdots \to_{\mathcal{A}} s_k$ such that $s_0 \in P$ and $s_k \in E \subseteq NF_{\mathcal{A}}$. Since $P \Rightarrow Q$ is a safety APR predicate for E, we have that $Q \cap E = \emptyset$, $\{t \in NF_{\mathcal{A}} \setminus E \mid \exists s \in P.\ s \to^*_{\mathcal{A}} t\} \subseteq Q$, and $Q \subseteq NF_{\mathcal{A}}$, and hence $\{s_0, s_1, \ldots, s_k\} \cap Q = \emptyset$. This contradicts the assumption that $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$. The *if* part can be proved analogously to the *only-if* part [18]. ◀

▶ **Theorem 5.4.** *Let $P \Rightarrow Q$ be a safety APR predicate for $E \subseteq NF_{\mathcal{A}}$. Then, $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$ if and only if $(A \cup \{\mathsf{any}\}, \to_{\mathcal{A}} \cup \{(s, \mathsf{any}) \mid s \in A \setminus E\}) \models^{\forall}_{\text{partial}} P \Rightarrow \{\mathsf{any}\}$.*

Proof. Let $\mathcal{A}' = (A \cup \{\mathsf{any}\}, \to_{\mathcal{A}'})$, where $\to_{\mathcal{A}'} = (\to_{\mathcal{A}} \cup \{(s, \mathsf{any}) \mid s \in A \setminus E\})$. Then, we have that $NF_{\mathcal{A}'} = E \cup \{\mathsf{any}\}$. We prove the *only-if* part by contradiction. Assume that $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$ and $\mathcal{A}' \not\models^{\forall}_{\text{partial}} P \Rightarrow \{\mathsf{any}\}$. Then, there exists a finite execution path $s_0 \to_{\mathcal{A}'} s_1 \to_{\mathcal{A}'} \cdots \to_{\mathcal{A}'} s_k \in E$ with $\mathsf{any} \notin \{s_0, s_1, \ldots, s_k\}$, and thus $s_0 \to_{\mathcal{A}} s_1 \to_{\mathcal{A}} \cdots \to_{\mathcal{A}} s_k \in E$. It follows from Theorem 5.2 that $\mathcal{A} \not\models^{\forall}_{\text{partial}} P \Rightarrow Q$. This contradicts the assumption that $\mathcal{A} \models^{\forall}_{\text{partial}} P \Rightarrow Q$. The *if* part can be proved analogously to the *only-if* part [18]. ◀

▶ **Theorem 6.3.** *Let $((V, \mathrm{a}, \mathrm{r}, c), \xi)$ be a partially valid APR proof for an APR predicate $P \Rightarrow Q$ w.r.t. $\mathcal{A}$, and (V', E) be the proof graph of the APR proof. Then, (V', E) is acyclic if and only if $\mathcal{A} \models^{\forall}_{\text{total}} P \Rightarrow Q$.*

Proof. We first prove the *only-if* part by contradiction. Assume that (V', E) is acyclic and $\mathcal{A} \not\models^{\forall}_{\text{total}} P \Rightarrow Q$. Then, there exists a (possibly infinite) execution path $s_0 \to_{\mathcal{A}} s_1 \to_{\mathcal{A}} \cdots$ such that $s_0 \in P$ and none of $s_0, s_1, \ldots$ is in Q. We make a case analysis depending on whether the path is finite.

- Case where the execution path is finite. Let the path have n steps, i.e., $s_0 \to_{\mathcal{A}} s_1 \to_{\mathcal{A}} \cdots \to_{\mathcal{A}} s_n \in NF_{\mathcal{A}}$. Since $s_n \in NF_{\mathcal{A}} \setminus Q$, there exists a node $v \in V$ such that $s_n \in a_L(v) \subseteq A$, and thus, $r(v) = \textsc{Dis}$. This contradicts the assumption.
- Case where the execution path is infinite. Since none of $s_0, s_1, \ldots$ is in Q, there exists an infinite path of the proof graph (V', E). Since $((V, \mathrm{a}, \mathrm{r}, c), \xi)$ is an APR proof, $(V, \mathrm{a}, \mathrm{r}, c)$ is a finite derivation tree, and thus (V', E) is a finite graph having an infinite path. Thus, (V', E) has a cycle. This contradicts the assumption.

Next, we prove the *if* part by contradiction. Assume that $\mathcal{A} \models^{\forall}_{\text{total}} P \Rightarrow Q$ and (V', E) has a cycle. Then, there exists a bud node v and its companion v' in the cycle such that $\xi(v) = v'$. By the assumption, we have that $\mathrm{a}(v) = \mathrm{a}(v')$, and thus $a_L(v) = a_L(v')$. Since v, v' are in the cycle, there exists a finite path from v' to v: $v' \to_{\textsc{Der}} v_1 \, (\to_{\textsc{Der}} \cup \to_{\textsc{Subs}})^* \, v$. It follows from Proposition 4.6 (2)–(3) that for any element $s \in a_L(v)$, there exists an element $s' \in a_L(v')$ such that $s' \to^+_{\mathcal{A}} s$ and the reduction has no element in Q. It follows from $a_L(v) = a_L(v')$ that for any element $s \in a_L(v)$, there exists an element $s' \in a_L(v)$ such that $s' \to^+_{\mathcal{A}} s$ and the reduction has no element in Q. Thus, there exists an infinite execution path that starts with an element $s'' \in a_L(v)$ and does not include any element in Q. In the same way, we have a reduction sequence $s_0 \to^*_{\mathcal{A}} s''$ such that $s_0 \in P$, and thus there exists an infinite execution path that starts with an element $s'' \in a_L(v)$ and does not include any element in Q. This contradicts the assumption. ◀