EasyChair Preprint

№ 8671

On Reducing Non-Occurrence of Specified Runtime Errors to All-Path Reachability Problems of Constrained Rewriting

Misaki Kojima and Naoki Nishida

August 12, 2022

On Reducing Non-Occurrence of Specified Runtime Errors to All-Path Reachability Problems of Constrained Rewriting*

Misaki Kojima
Nagoya University, Nagoya, Japan
k-misaki@trs.css.i.nagoya-u.ac.jp

Naoki Nishida
Nagoya University, Nagoya, Japan
nishida@i.nagoya-u.ac.jp

A concurrent program with semaphore-based exclusive control can be transformed into a logically constrained term rewrite system that is computationally equivalent to the program. In this paper, we first propose a framework to reduce the non-occurrence of a specified runtime error in the program to an all-path reachability problem of the transformed logically constrained term rewrite system. Here, an all-path reachability problem is a pair of state sets and is demonically valid if every finite execution path starting with a state in the first set and ending with a terminating state includes a state in the second set. Then, as a case study, we show how to apply the framework to the race-freeness of semaphore-based exclusive control in the program. Finally, we show a simplified variant of a proof system for all-path reachability problems.

1 Introduction

Recently, approaches to program verification by means of logically constrained term rewrite systems (LCTRSs, for short) [10] are well investigated [5, 18, 4, 11, 6, 7]. LCTRSs are useful as computation models of not only functional but also imperative programs. For instance, equivalence checking by means of LCTRSs is useful to ensure correctness of terminating functions (cf. [5]). Here, equivalence of two functions means that for every input, the functions return the same output or end with the same projection of final configurations. In previous work [9], the method of transforming sequential programs into LCTRSs has been extended to concurrent programs with semaphore-based exclusive control. In the extension, each rewrite rule represents an action of exactly one process, and any list structure is not used to represent waiting queues for semaphores.

In previous work [5, 6, 7, 9], to verify correctness of programs, *rewriting induction* (RI, for short) [13, 5] is used to prove equivalence of two programs. Since RI requires termination of LCTRSs, it cannot be used to verify non-terminating programs. Furthermore, it is difficult (or impossible) to reduce the non-occurrence of a specified runtime-error to equivalence, i.e., an *inductive theorem* which is a valid equation of terms w.r.t. the reduction of a given LCTRS. This is because for an equation being an inductive theorem, the notion requires the *existence* of a reduction sequence between instances of the LHS and RHS of the equation, i.e., *not all* reduction sequences between the two instances are taken into account. To take into account *all* reduction sequences, we focus on *all-path reachability* [14, 4, 15] which may take the place of equivalence. An *all-path reachability problem* (APR problem, for short) of a rewrite system is a pair $P \Rightarrow Q$ of state sets P, Q and is *demonically valid* if every finite execution path—a reduction sequence starting with a state in P and ending with a terminating state—includes a state in Q. DCC, a proof system for APR problems of LCTRSs, has been proposed in [4] and APR problems must be useful for verification by means of LCTRSs. In addition, target LCTRSs do not have to be terminating.

*This work was partially supported by JSPS KAKENHI Grant Number 18K11160 and DENSO Corporation.

Submitted to:
WPTE 2022

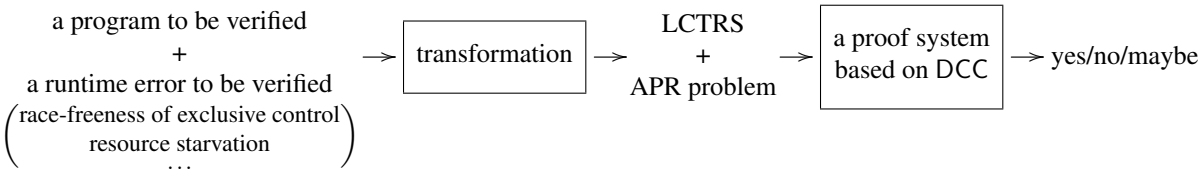

Figure 1: our framework for runtime-error verification

In this paper, we aim at developing a method for runtime-error verification by means of LCTRSs. To this end, we first propose a framework to reduce the non-occurrence of a specified runtime error in a concurrent program to an APR problem of the transformed LCTRS (Section 4.1). Then, as a case study, we show how to apply the framework to the race-freeness of semaphore-based exclusive control (Section 4.2). Finally, we show a simplified variant of the proof system DCC (Section 5). Fig. 1 illustrates our framework for runtime-error verification. Missing proofs can be seen in the appendix.

In addition to the LCTRS obtained from a concurrent program, the framework takes as an input a runtime error specified by a finite set of constrained terms representing error states. In the framework, we introduce two fresh constants, success and error, and add two kinds of rewrite rules into the LCTRS: One reduces any state to success, and the other rewrites all instances of the constrained terms to error. Then, the framework outputs the APR problem $\{\text{initial state(s)}\} \Rightarrow \{\mathsf{success}\}$, together with the updated LCTRS. By the definition of all-path reachability and the added rewrite rules, it holds that the generated APR problem is true if and only if there is no reduction sequence of the original LCTRS from initial states to any error state.

The contribution of this paper is to show an application of APR problems to runtime-error verification. Such an application must increase the usefulness and practicability of LCTRSs.

Related Work The proof system DCC for LCTRSs has been proposed in [4] to prove reachability properties, especially partial correctness of transition systems described by LCTRSs. In [3], total correctness of such transition systems is reduced to partial correctness (i.e., APR problems) by means of a transformation of the language semantics. On the other hand, the property to be verified in this paper is the non-occurrence of specified runtime errors, and using some characteristics of the property, we propose a simplified variant of DCC.

2 Preliminaries

In this section, we briefly recall logically constrained rewriting [10, 5] and all-path reachability [14, 4, 15]. Familiarity with basic notions on term rewriting [1, 12] is assumed.

2.1 Logically Constrained Rewriting

Let $\mathcal{S}$ be a set of *sorts* and $\mathcal{V}$ a (countably infinite) set of *variables*, each of which is equipped with a sort. A *signature* Σ disjoint from $\mathcal{V}$ is a set of *function symbols* f, each of which is equipped with a *sort declaration* $\iota_1 \times \cdots \times \iota_n \Rightarrow \iota$, written as $f : \iota_1 \times \cdots \times \iota_n \Rightarrow \iota$, where $\iota_1, \ldots, \iota_n, \iota \in \mathcal{S}$. In the rest of this section, we fix $\mathcal{S}$, Σ, and $\mathcal{V}$ and use them without notice in the paper. We denote the set of well-sorted *terms* over Σ and $\mathcal{V}$ by $T(\Sigma, \mathcal{V})$. We may write $s : \iota$ if s has sort ι. The set of variables occurring in $s_1, \ldots, s_n$ is denoted by $\mathcal{V}ar(s_1, \ldots, s_n)$. Given a term t and a *position* p (a sequence of positive integers)

of t, $t|_p$ denotes the subterm of t at position p, and $s[t]_p$ denotes s with the subterm at position p replaced by t, where the sorts of $s|_p$ and t coincide.

A *substitution* γ is a sort-preserving total mapping from $\mathcal{V}$ to $T(\Sigma,\mathcal{V})$, and naturally extended for a mapping from $T(\Sigma,\mathcal{V})$ to $T(\Sigma,\mathcal{V})$. The *domain* $\mathcal{D}om(\gamma)$ of γ is the set of variables x with $\gamma(x)\neq x$, and the *range* of γ is denoted by $\mathcal{R}an(\gamma)$. The restriction of γ w.r.t. a set X of variables is denoted by $\gamma|_X$: $\gamma|_X(x)=\gamma(x)$ if $x\in X$, and otherwise $\gamma|_X(x)=x$. The application of γ to term s is denoted by $s\gamma$.

To define LCTRSs, we consider the following signatures, mappings, and constants: Two signatures Σ_{terms} and Σ_{theory} such that $\Sigma=\Sigma_{terms}\cup\Sigma_{theory}$; a mapping $\mathcal{I}$ that assigns to each sort ι occurring in Σ_{theory} a set $\mathcal{I}_\iota$, i.e., $\mathcal{I}(\iota)=\mathcal{I}_\iota$; a mapping $\mathcal{J}$ that assigns to each $f:\iota_1\times\cdots\times\iota_n\Rightarrow\iota\in\Sigma_{theory}$ a function $f^{\mathcal{J}}$ in $\mathcal{I}_{\iota_1}\times\cdots\times\mathcal{I}_{\iota_n}\Rightarrow\mathcal{I}_\iota$, i.e., $\mathcal{J}(f)=f^{\mathcal{J}}$; a set $\mathcal{V}al_\iota\subseteq\Sigma_{theory}$ of *value-constants* $a:\iota$ for each sort ι occurring in Σ_{theory} such that $\mathcal{J}$ gives a bijective mapping from $\mathcal{V}al_\iota$ to $\mathcal{I}_\iota$. Note that for each sort, $\mathcal{I}$ specifies the universe, and for each symbol, $\mathcal{J}$ specifies the interpretation. We denote $\bigcup_{\iota\in\mathcal{S}}\mathcal{V}al_\iota$ by $\mathcal{V}al$. We require that $\Sigma_{terms}\cap\Sigma_{theory}\subseteq\mathcal{V}al$. The sorts occurring in Σ_{theory} are called *theory sorts*, and the symbols *theory symbols*. Symbols in $\Sigma_{theory}\setminus\mathcal{V}al$ are *calculation symbols*. A term in $T(\Sigma_{theory},\mathcal{V})$ is called a *theory term*. For ground theory terms, we define the *interpretation* $[\![\cdot]\!]$ as $[\![f(s_1,\ldots,s_n)]\!]=\mathcal{J}(f)([\![s_1]\!],\ldots,[\![s_n]\!])$. Note that for every ground theory term s, there is a unique value-constant c such that $[\![s]\!]=[\![c]\!]$. We may use infix notation for calculation symbols.

We typically choose a theory signature with $\Sigma_{theory}\supseteq\Sigma^{core}_{theory}$, where Σ^{core}_{theory} includes *bool*, a sort of *Booleans*, such that $\mathcal{V}al_{bool}=\{\mathsf{true},\mathsf{false}\}$ and $\mathcal{I}(bool)=\{\top,\bot\}$, $\Sigma^{core}_{theory}=\mathcal{V}al_{bool}\cup\{\wedge,\vee,\Longrightarrow:bool\times bool\Rightarrow bool,\ \neg:bool\Rightarrow bool\}\cup\{=_\iota,\neq_\iota:\iota\times\iota\Rightarrow bool\mid\iota\text{ is a theory sort in }\Sigma_{theory}\}$, and $\mathcal{J}$ interprets these symbols as expected: $\mathcal{J}(\mathsf{true})=\top$ and $\mathcal{J}(\mathsf{false})=\bot$. We omit the sort subscripts from $=$ and $\neq$ when they are clear from context. The standard integer signature Σ^{int}_{theory} is $\Sigma^{core}_{theory}\cup\{+,-,\times,\mathsf{exp},\mathsf{div},\mathsf{mod}:int\times int\Rightarrow int\}\cup\{\geq,>:int\times int\Rightarrow bool\}\cup\mathcal{V}al_{int}$ where $\mathcal{S}\supseteq\{int,bool\}$, $\mathcal{V}al_{int}=\{\mathsf{n}:int\mid n\in\mathbb{Z}\}$, $\mathcal{I}(int)=\mathbb{Z}$, and $\mathcal{J}(\mathsf{n})=n$. Note that we use n (in sans-serif font) as the function symbol for $n\in\mathbb{Z}$ (in *math* font). We define $\mathcal{J}$ in the natural way.

A *constrained rewrite rule* is a triple $\ell\to r\ [\phi]$ such that ℓ and r are terms of the same sort, ϕ is a constraint, and ℓ has the form $f(\ell_1,\ldots,\ell_n)$ that is not a theory term. If $\phi=\mathsf{true}$, then we may write $\ell\to r$. We define $\mathcal{LV}ar(\ell\to r\ [\phi])$ as $\mathcal{V}ar(\phi)\cup(\mathcal{V}ar(r)\setminus\mathcal{V}ar(\ell))$. We say that a substitution γ *respects* $\ell\to r\ [\phi]$ if $\mathcal{R}an(\gamma|_{\mathcal{LV}ar(\ell\to r\ [\phi])})\subseteq\mathcal{V}al$ and $[\![\phi\gamma]\!]=\top$. Note that it is allowed to have $\mathcal{V}ar(r)\not\subseteq\mathcal{V}ar(\ell)$, but fresh variables in the right-hand side may only be instantiated with *value-constants* (see the definition of $\to_{\mathcal{R}}$ below). We let $\mathcal{R}_{calc}$ be the set $\{f(x_1,\ldots,x_n)\to y\ [y=f(x_1,\ldots,x_n)]\mid f\in\Sigma_{theory}\setminus\mathcal{V}al,\ x_1,\ldots,x_n,y\in\mathcal{V}\}$. The elements of $\mathcal{R}_{calc}$ are also called constrained rewrite rules (or *calculation rules*) even though their left-hand side is a theory term. The *rewrite relation* $\to_{\mathcal{R}}$ is a binary relation on terms, defined as follows: for a term s, $s[\ell\gamma]_p\to_{\mathcal{R}}s[r\gamma]_p$ if $\ell\to r\ [\phi]\in\mathcal{R}\cup\mathcal{R}_{calc}$ and γ respects $\ell\to r\ [\phi]$.

Now we define a *logically constrained term rewrite system* (LCTRS, for short) as an abstract reduction system $(T(\Sigma,\mathcal{V}),\to_{\mathcal{R}})$ where $\mathcal{R}$ is a set of constrained rewrite rules. LCTRS $(T(\Sigma,\mathcal{V}),\to_{\mathcal{R}})$ is simply denoted by $\mathcal{R}$. An LCTRS is usually given by supplying Σ, $\mathcal{R}$, and an informal description of $\mathcal{I}$ and $\mathcal{J}$ if these are not clear from context. The set of *normal forms* of $\mathcal{R}$ is denoted by $NF_{\mathcal{R}}$. LCTRS $\mathcal{R}$ is said to be *terminating* (*confluent*) if $\to_{\mathcal{R}}$ is terminating (confluent).

Example 2.1 Let $\mathcal{S}=\{int,bool\}$, $\Sigma=\Sigma_{terms}\cup\Sigma^{int}_{theory}$ and $\Sigma_{terms}=\{\mathsf{fact}:int\Rightarrow int\}\cup\{\mathsf{n}:int\mid n\in\mathbb{Z}\}$. We reduce $3-1$ to 2 in one step with the calculation rule $x-y\to z\ [z=x-y]$, and $3\times(2\times(1\times1))$ to $\mathsf{6}$ in three steps. To implement an LCTRS calculating the *factorial* function over $\mathbb{Z}$, we use the signature Σ above and the LCTRS $\mathcal{R}_{fact}=\{\ \mathsf{fact}(x)\to1\ [x\leq0]\quad\mathsf{fact}(x)\to x\times\mathsf{fact}(x-1)\ [\neg(x\leq0)]\ \}$. The term $\mathsf{fact}(3)$ is reduced by $\mathcal{R}_{fact}$ to $\mathsf{6}$: $\mathsf{fact}(3)\to_{\mathcal{R}_{fact}}3\times\mathsf{fact}(3-1)\to_{\mathcal{R}_{fact}}3\times\mathsf{fact}(2)\to_{\mathcal{R}_{fact}}\cdots\to_{\mathcal{R}_{fact}}\mathsf{6}$.

A *constrained term* is a pair $\langle t\mid\phi\rangle$ of a term t and a constraint ϕ. The set of all ground instances

of $\langle t \mid \phi \rangle$ is denoted by $[\![\langle t \mid \phi \rangle]\!]$: $[\![\langle t \mid \phi \rangle]\!] = \{t\gamma \mid \mathcal{D}om(\gamma) \supseteq \mathcal{V}ar(t),\ \mathcal{R}an(\gamma) \subseteq T(\Sigma),\ \gamma \text{ respects } \phi\}$. This paper considers constrained terms as sets of ground terms. A constrained equation $s \approx t\ [\phi]$ is called an *inductive theorem* of an LCTRS $\mathcal{R}$ if $s\gamma \leftrightarrow^*_{\mathcal{R}} t\gamma$ for every ground substitution γ such that $\mathcal{D}om(\gamma) \supseteq \mathcal{V}ar(s,t,\phi)$ and γ respects ϕ.

2.2 All-Path Reachability

Let $(M, \rightharpoonup)$ be a transition system with $\rightharpoonup \subseteq M \times M$. An *execution path* is either an infinite $\rightharpoonup$-sequence or a finite $\rightharpoonup$-sequence ending with an irreducible term. A *state predicate* is a set P ($\subseteq M$). The predicate P is said to be *runnable* (w.r.t. $\rightharpoonup$) if $P \neq \emptyset$ and for each $t \in P$ there exists $t' \in M$ such that $t \rightharpoonup t'$. Note that a runnable state predicate does not include any irreducible state. A *reachability predicate* (w.r.t. $(M, \rightharpoonup)$) is a pair $P \Rightarrow Q$ of state predicates P, Q.

An execution path τ is said to *satisfy* a reachability predicate $P \Rightarrow Q$, written $\tau \models^{\forall} P \Rightarrow Q$, if τ starts with a state in P and τ includes a state in Q, whenever τ is finite. A reachability predicate $P \Rightarrow Q$ is said to be *demonically valid* (w.r.t. $(M, \rightharpoonup)$) if $\tau \models^{\forall} P \Rightarrow Q$ for all execution paths τ starting from a state in P. The demonical validity of $P \Rightarrow Q$ means that every finite execution path starting from a state in P eventually reaches a state in Q.

Let $\mathcal{R}$ be an LCTRS and $\langle t_\ell \mid \phi_\ell \rangle, \langle t_r \mid \phi_r \rangle$ constrained terms. We call the pair $\langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle t_r \mid \phi_r \rangle$ an *all-path reachability problem* (APR problem, for short) of $\mathcal{R}$. Note that $\langle t_\ell \mid \phi_\ell \rangle$ and $\langle t_r \mid \phi_r \rangle$ may have shared variables. The *set of derivatives* of a constrained term $\langle t \mid \phi \rangle$ is defined as follows:

$$\Delta_{\mathcal{R}}(\langle t \mid \phi \rangle) = \bigcup_{\ell \to r\ [\psi] \in \mathcal{R}} \Delta_{\ell,r,\psi}(\langle t \mid \phi \rangle)$$

where $\ell \to r\ [\psi]$ has no shared variable with $\langle t \mid \phi \rangle$[1] and $\Delta_{\ell,r,\psi}(\langle t \mid \phi \rangle) = \{\langle (t[r]_p)\gamma \mid (\phi \wedge \psi)\gamma \rangle \mid p$ is a position of t, $t|_p$ and ℓ are unifiable, γ is a most general unifier of $t|_p$ and ℓ, $\mathcal{R}an(\gamma|_{\mathcal{V}ar(\phi,\psi)}) \subseteq \mathcal{V}al \cup \mathcal{V}$, $(\phi \wedge \psi)\gamma$ is satisfiable$\}$. A constrained term $\langle t \mid \phi \rangle$ is said to be *$\mathcal{R}$-derivable* if $\Delta_{\mathcal{R}}(\langle t \mid \phi \rangle) \neq \emptyset$. We say that $\mathcal{R}$ *demonically satisfies* $\langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle t_r \mid \phi_r \rangle$, written $\mathcal{R} \models^{\forall} \langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle t_r \mid \phi_r \rangle$, if $[\![\langle t_\ell\gamma \mid \phi_\ell\gamma \rangle]\!] \Rightarrow [\![\langle t_r\gamma \mid \phi_r\gamma \rangle]\!]$ is demonically valid w.r.t. $(T(\Sigma), \to_{\mathcal{R}})$ for any ground substitution γ such that $\mathcal{D}om(\gamma) = \mathcal{V}ar(t_\ell, \phi_\ell) \cap \mathcal{V}ar(t_r, \phi_r)$, and $\mathcal{R}an(\gamma) \subseteq T(\Sigma)$, and $\mathcal{R}an(\gamma|_{\mathcal{V}ar(\phi_\ell) \cap \mathcal{V}ar(\phi_r)}) \subseteq \mathcal{V}al$.

3 LCTRSs Modeling Programs with Semaphores

In this section, using Program 1, we briefly illustrate how to model by an LCTRS a concurrent programs with exclusive control operated by means of semaphores.

A semaphore is a variable to specify the availability of a corresponding shared resource. Note that the range of semaphores are non-negative integers from 0 to some specified upper limit u: $u = 1$ for *binary* semaphores, and $u > 1$ for *counting* ones. This paper deals with the operations `down` and `up` of a semaphore `s` for a shared resource *src*, which are defined as follows [17]:

- Operation `down` is executed as `down(&s)` before accessing *src*: If the value of `s` is not 0, then it is decreased by 1; otherwise, the process executing `down` goes into the wait state, queuing up for `s`.
- Operation `up` is executed as `up(&s)` in releasing the access to *src*: If at least one process waits for `s`, then the value of `s` is kept as it is and the process at the top of the waiting queue goes into the executable state;[2] otherwise, the value of `s` is increased by 1.

[1] When there exists a shared variable, we rename the variables in $\ell \to r\ [\psi]$.

[2] The process going into the executable state acquires the semaphore `s` and starts to access the shared resource.

Program 1: a simple reader-writer program with a counting semaphore

```
1  semaphore s = 2;
2  int x = 0;
3
4  void reader(void)
5  {
6    while(true){
7      int y;
8
9      down(&s);
10     y = x;
11     up(&s);
12   }
13 }
14
15 void writer(void)
16 {
17   while(true){
18     down(&s);
19     x = 1;
20     up(&s);
21   }
22 }
```

- `down` and `up` are *atomic operations*—when an atomic operation a is executed, any other operations (processes) cannot interrupt until the execution of a finishes.

Note that the above specification can perform for both binary and counting semaphores.

Let us consider a configuration with three processes, the first and second of which execute `reader` in Program 1, and the third of which executes `writer` in Program 1. For such a configuration, the LCTRS $\mathcal{R}_1$ in Fig. 2 with the following initial term is generated [9]:

$$\mathsf{cnfg}(\ \underbrace{\mathsf{p}(\mathsf{rdr}_7,0)}_{\text{1st process}},\ \underbrace{\mathsf{p}(\mathsf{rdr}_7,0)}_{\text{2nd process}},\ \underbrace{\mathsf{p}(\mathsf{wtr}_{18},0)}_{\text{3rd process}},\ \underbrace{\mathsf{sem}(2,1,3)}_{\mathtt{s}},\ \underbrace{0}_{\mathtt{x}},\ \underbrace{0}_{\text{flag for atomic operation}}\ )$$

where $\mathsf{cnfg} : \mathit{process} \times \mathit{process} \times \mathit{process} \times \mathit{semaphore} \times \mathit{int} \times \mathit{int} \to \mathit{cnfg}$ $\mathsf{p} : \mathit{fcall} \times \mathit{int} \to \mathit{process}$ $\mathsf{sem} : \mathit{int} \times \mathit{int} \times \mathit{int} \to \mathit{semaphore}$ The i-th argument with $1 \le i \le 3$ is the i-th process; the fourth is a semaphore $\mathsf{sem}(v_{\mathtt{s}},d,t)$ with $v_{\mathtt{s}}$ a stored value, d a display to call a waiting process, and t a next ticket issued for an upcoming waiting process; the fifth is a value stored in `x`; the sixth is a flag for atomic operations represented by two or more rewrite rules.[3] Unlike [9], this paper uses integers instead of bit vectors for `int`. For page limitation, we omit the detailed explanation for the construction (see [9] for detail).

4 Reducing Non-Occurrence of a Runtime-Error to an APR Problem

In this section, to verify the non-occurrence of a specified runtime error, we first propose a framework to reduce it to an APR problem. Then, as a case study, we show how to apply the framework to the race-freeness of semaphore-based exclusive control.

All-path reachability $P \Rightarrow Q$ w.r.t. a transition system is a property that for every *finite* execution path starting from P, its initial state reaches a state in Q. This means that no *infinite* execution path is taken into account for the transition system having the property. On the other hand, the non-occurrence of a runtime error specified by error states has to take into account all execution paths, i.e., not only finite but also *infinite* ones, because error states may occur in an infinite execution path. For this reason, we need a breakthrough to take into account infinite execution paths in the APR setting where infinite paths are excluded. To fill the gap, we do not modify the ordinary APR setting—the definition and proof system of APR problems—and use the setting as it is.

[3] For non-terminating LCTRSs, atomic operations represented by two or more rewrite rules without this flag not in [9] may be stuck because of another non-terminating process. To avoid such stuck, this flag has been introduced [8].

$$\left\{\begin{array}{ll}
\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_7,n),p_2,p_3,\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_9(0),n),p_2,p_3,\mathsf{sem}(s,d,t),x,0) & \\
\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_9(y),0),p_2,p_3,\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{10}(y),0),p_2,p_3,\mathsf{sem}(s-1,d,t),x,0) & [s\neq 0]\\
\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_9(y),0),p_2,p_3,\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_9(y),t),p_2,p_3,\mathsf{sem}(s,d,t+2),x,0) & [s=0]\\
\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_9(y),n),p_2,p_3,\mathsf{sem}(s,d,t),x,1)\to\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{10}(y),0),p_2,p_3,\mathsf{sem}(s,d,t),x,0) & [n=d\wedge n\neq 0]\\
\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{10}(y),n),p_2,p_3,\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{11}(x),n),p_2,p_3,\mathsf{sem}(s,d,t),x,0) & \\
\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{11}(y),0),p_2,p_3,\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_7,0),p_2,p_3,\mathsf{sem}(s,d+2,t),x,1) & [t\neq d+2]\\
\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{11}(y),0),p_2,p_3,\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_7,0),p_2,p_3,\mathsf{sem}(s+1,d,t),x,0) & [t=d+2]\\
\mathsf{cnfg}(p_1,\mathsf{p}(\mathsf{rdr}_7,n),p_3,\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(p_1,\mathsf{p}(\mathsf{rdr}_9(0),n),p_3,\mathsf{sem}(s,d,t),x,0) & \\
\vdots & \\
\mathsf{cnfg}(p_1,\mathsf{p}(\mathsf{rdr}_{11}(y),0),p_3,\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(p_1,\mathsf{p}(\mathsf{rdr}_7,0),p_3,\mathsf{sem}(s+1,d,t),x,0) & [t=d+2]\\
\mathsf{cnfg}(p_1,p_2,\mathsf{p}(\mathsf{wtr}_{18},0),\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(p_1,p_2,\mathsf{p}(\mathsf{wtr}_{19},0),\mathsf{sem}(s-1,d,t),x,0) & [s\neq 0]\\
\mathsf{cnfg}(p_1,p_2,\mathsf{p}(\mathsf{wtr}_{18},0),\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(p_1,p_2,\mathsf{p}(\mathsf{wtr}_{18},t),\mathsf{sem}(s,d,t+2),x,0) & [s=0]\\
\mathsf{cnfg}(p_1,p_2,\mathsf{p}(\mathsf{wtr}_{18},n),\mathsf{sem}(s,d,t),x,1)\to\mathsf{cnfg}(p_1,p_2,\mathsf{p}(\mathsf{wtr}_{19},0),\mathsf{sem}(s,d,t),x,0) & [n=d\wedge n\neq 0]\\
\mathsf{cnfg}(p_1,p_2,\mathsf{p}(\mathsf{wtr}_{19},n),\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(p_1,p_2,\mathsf{p}(\mathsf{wtr}_{20},n),\mathsf{sem}(s,d,t),1,0) & \\
\mathsf{cnfg}(p_1,p_2,\mathsf{p}(\mathsf{wtr}_{20},0),\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(p_1,p_2,\mathsf{p}(\mathsf{wtr}_{18},0),\mathsf{sem}(s,d+2,t),x,1) & [t\neq d+2]\\
\mathsf{cnfg}(p_1,p_2,\mathsf{p}(\mathsf{wtr}_{20},0),\mathsf{sem}(s,d,t),x,0)\to\mathsf{cnfg}(p_1,p_2,\mathsf{p}(\mathsf{wtr}_{18},0),\mathsf{sem}(s+1,d,t),x,0) & [t=d+2]
\end{array}\right\}$$

Figure 2: the transformed LCTRS $\mathcal{R}_1$ for Program 1

Our approach to the gap is to make all finite prefixes of infinite execution paths be execution paths, i.e., to append a terminating state to all transition sequences. To be more precise, given an LCTRS $\mathcal{R}$ as a transition system, we introduce to $\mathcal{R}$ a fresh constant $\mathsf{success}$ and a rewrite rule $\mathsf{cnfg}(\ldots)\to\mathsf{success}$ that reduces all states to $\mathsf{success}$. Such a constant and a rewrite rule imply that for an infinite sequence $t_1\to_{\mathcal{R}} t_2\to_{\mathcal{R}}\cdots$ and every finite prefix sequence $t_1\to_{\mathcal{R}} t_2\to_{\mathcal{R}}\cdots\to_{\mathcal{R}} t_n$, all states $t_1,\ldots,t_n$ are included in an execution path $t_1\to_{\mathcal{R}} t_2\to_{\mathcal{R}}\cdots\to_{\mathcal{R}} t_n\to_{\mathcal{R}}\mathsf{success}$.

To ensure the non-occurrence of error states, we consider an APR problem $\langle s_0\mid\phi_0\rangle\Rightarrow\langle\mathsf{success}\mid\mathsf{true}\rangle$. Since every state can reach $\mathsf{success}$, we make error states not reach $\mathsf{success}$ by introducing to $\mathcal{R}$ another fresh constant error and rewrite rules that reduce all error states to error. The occurrence of an error state in an execution path implies the existence of a finite execution path that does not include $\mathsf{success}$.

4.1 A Framework of Reduction to APR Problems

Let P be a program to be verified, $\mathcal{R}$ an LCTRS obtained by transforming P, and $\langle s_0\mid\phi_0\rangle$ a constrained term representing the initial state of P.[4] In addition, let $\{\langle u_i\mid\phi_i\rangle\mid 1\leq i\leq n\}$ be a finite set of constrained terms representing the states, called *error states*, of a certain run-time error to be verified. Introducing fresh constants $\mathsf{success}$ and error with sort *cnfg* into the signature Σ, we reduce the non-occurrence of the runtime error specified by $\{\langle u_i\mid\phi_i\rangle\mid 1\leq i\leq n\}$ to the APR problem $\langle s_0\mid\phi_0\rangle\Rightarrow\langle\mathsf{success}\mid\mathsf{true}\rangle$ of the LCTRS $\mathcal{R}^{\forall}$ that is defined as follows:

$$\mathcal{R}^{\forall}:=\mathcal{R}\cup\{\,\mathsf{cnfg}(p_1,\ldots,p_m,x_1,\ldots,x_k,x_a)\to\mathsf{success}\,\}\cup\{\,u_i\to\mathsf{error}\ [\phi_i]\mid 1\leq i\leq n\,\}$$

where $p_1,\ldots,p_m$ are variables for processes, $x_1,\ldots,x_k$ are variables for semaphores and global variables, and x_a is a variable for the flag for atomic operations represented by two or more rewrite rules.

Rule $\mathsf{cnfg}(p_1,\ldots,p_m,x_1,\ldots,x_k,x_a)\to\mathsf{success}$ rewrites all states to $\mathsf{success}$. Each rule $u_i\to\mathsf{error}\ [\phi_i]$ rewrites all states represented by $\langle u_i\mid\phi_i\rangle$ to error. Note that every finite execution path of $\mathcal{R}^{\forall}$ starting from $[\![\langle s_0\mid\phi_0\rangle]\!]$ ends with either $\mathsf{success}$ or error.

[4] $[\![\langle s_0\mid\phi_0\rangle]\!]$ do not have to be a singleton set.

The demonical validity of $\langle s_0 \mid \phi_0 \rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true} \rangle$ w.r.t. $\mathcal{R}^\forall$ is equivalent to the non-existence of an execution path of $\mathcal{R}^\forall$, that includes error. This is because there is no rule that rewrites either error or success, and thus, error cannot be rewritten to success, and vice versa. Therefore, by proving $\langle s_0 \mid \phi_0 \rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true} \rangle$ to be demonically valid, we can ensure the non-existence of a finite rewrite sequence of $\mathcal{R}$ that starts with a ground term in $[\![\langle s_0 \mid \phi_0 \rangle]\!]$ and includes a ground term in $[\![\langle u_i \mid \phi_i \rangle]\!]$ for some i, that is, the runtime error to be verified does not occur in any execution of P.

Theorem 4.1 *$\mathcal{R}^\forall \models^\forall \langle s_0 \mid \phi_0 \rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true} \rangle$ if and only if $s \not\to^*_{\mathcal{R}} u$ for any ground term $s \in [\![\langle s_0 \mid \phi_0 \rangle]\!]$ and any ground term $u \in \bigcup_{i=1}^{n} [\![\langle u_i \mid \phi_i \rangle]\!]$.*

Any error state has two normal forms, success and error, and $\mathcal{R}^\forall$ is not ground confluent. The normal forms of $\mathcal{R}^\forall$ with sort *cnfg* are success and error. By definition, it is clear that $\langle s_0 \mid \phi_0 \rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true} \rangle$ is demonically valid w.r.t. $\mathcal{R}^\forall$ if and only if every ground term in $[\![\langle s_0 \mid \phi_0 \rangle]\!]$ has a unique normal form. From this fact, for the demonical validity of our APR problems, we may prove $\mathrm{UN}^{\to}$ property of ground terms in $[\![\langle s_0 \mid \phi_0 \rangle]\!]$. Though, for the present, we have no method to prove such $\mathrm{UN}^{\to}$ property because e.g., $\mathcal{R}^\forall$ is not ground confluent.

4.2 A Case Study: Race-Freeness of Semaphore-based Exclusive Control

Let $\mathtt{s}_{\max}$ be the upper limit of a semaphore `s`. The exclusive control by means of `s` *fails* when $\mathtt{s}_{\max}$ or more processes are in their critical sections w.r.t. `s` simultaneously. A process is said to be *in the critical section* if the present state of the process is between the state just after getting the semaphore by means of `down` and the state just before executing `up` to return the semaphore. Viewed in this light, for an LCTRS obtained from a program, we can determine which function symbols represent critical sections, that is, we can prepare a finite set of configurations under the critical race condition.

Example 4.2 Consider the transformed LCTRS $\mathcal{R}_1$ in Fig. 2 and the following initial configuration for Program 1 again: $\mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_7,0),\mathsf{p}(\mathsf{rdr}_7,0),\mathsf{p}(\mathsf{wtr}_{18},0),\mathsf{sem}(2,1,3),0,0)$. Recall that the upper limit (initial value) of the semaphore `s` is 2 and there are three processes, the first and second of which execute `reader`, and the third of which executes `writer`. At most two processes can access the global variable `x`—can be in their critical sections w.r.t. `s`—simultaneously. A process executing `reader` is in the critical section when executing statements on Lines 10–11 in Program 1, and a process executing `writer` is in the critical section when executing statements on Lines 19–20. Therefore, the following eight constrained terms represent the critical race condition, i.e., the configurations where three processes are in their critical sections w.r.t. `s` simultaneously:

$$\begin{array}{l}
\langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{10}(y),n),\mathsf{p}(\mathsf{rdr}_{10}(y),n),\mathsf{p}(\mathsf{wtr}_{19},n),\mathit{sem},x,a) \mid \mathsf{true}\rangle\\
\langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{10}(y),n),\mathsf{p}(\mathsf{rdr}_{10}(y),n),\mathsf{p}(\mathsf{wtr}_{20},n),\mathit{sem},x,a) \mid \mathsf{true}\rangle\\
\langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{10}(y),n),\mathsf{p}(\mathsf{rdr}_{11}(y),n),\mathsf{p}(\mathsf{wtr}_{19},n),\mathit{sem},x,a) \mid \mathsf{true}\rangle\\
\langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{10}(y),n),\mathsf{p}(\mathsf{rdr}_{11}(y),n),\mathsf{p}(\mathsf{wtr}_{20},n),\mathit{sem},x,a) \mid \mathsf{true}\rangle\\
\langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{11}(y),n),\mathsf{p}(\mathsf{rdr}_{10}(y),n),\mathsf{p}(\mathsf{wtr}_{19},n),\mathit{sem},x,a) \mid \mathsf{true}\rangle\\
\langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{11}(y),n),\mathsf{p}(\mathsf{rdr}_{10}(y),n),\mathsf{p}(\mathsf{wtr}_{20},n),\mathit{sem},x,a) \mid \mathsf{true}\rangle\\
\langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{11}(y),n),\mathsf{p}(\mathsf{rdr}_{11}(y),n),\mathsf{p}(\mathsf{wtr}_{19},n),\mathit{sem},x,a) \mid \mathsf{true}\rangle\\
\langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{11}(y),n),\mathsf{p}(\mathsf{rdr}_{11}(y),n),\mathsf{p}(\mathsf{wtr}_{20},n),\mathit{sem},x,a) \mid \mathsf{true}\rangle
\end{array}$$

For $\mathcal{R}_1$, the initial configuration, and the above eight constrained terms, our framework generates the LCTRS $\mathcal{R}_1^\forall$ in Fig. 3 and the following APR problem of $\mathcal{R}_1$ to verify the race-freeness of exclusive

$$\mathcal{R}_1^{\forall} = \mathcal{R}_1 \cup \left\{ \begin{array}{r} \mathsf{cnfg}(p_1,p_2,p_3,\mathit{sem},x,f) \to \mathsf{success} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{10}(y),n),\mathsf{p}(\mathsf{rdr}_{10}(y),n),\mathsf{p}(\mathsf{wtr}_{19},n),\mathit{sem},x,a) \to \mathsf{error} \\ \vdots \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_{11}(y),n),\mathsf{p}(\mathsf{rdr}_{11}(y),n),\mathsf{p}(\mathsf{wtr}_{20},n),\mathit{sem},x,a) \to \mathsf{error} \end{array} \right\}$$

Figure 3: the LCTRS $\mathcal{R}_1^{\forall}$ generated by our framework of the reduction to APR problems.

control w.r.t. s in Program 3:

$$\langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_7,0),\mathsf{p}(\mathsf{rdr}_7,0),\mathsf{p}(\mathsf{wtr}_{18},0),\mathsf{sem}(2,1,3),0,0) \mid \mathsf{true}\rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true}\rangle$$

Another example—the *sleeping barber problem* [16]—to verify the race-freeness of semaphore-based exclusive control can be seen in the appendix.

5 A Simplified Proof System for APR Problems

In this section, we propose a simplified variant of the proof system DCC [4] for all-path reachability of ordinary LCTRSs and a certain class of APR problems.

In [4], LCTRSs have been extended by adding the *theory of equality* for non-theory symbols to the usual built-in theory specified by Σ_{theory}, $\mathcal{I}$, and $\mathcal{J}$. The proof system DCC for APR problems is intended for the extended LCTRSs. Structural-equivalence formulas such as $t_\ell = t_r$ take the place of *unification* used at case analysis steps of proving APR problems. Let us consider the following rule in DCC:

$$\frac{\langle t_\ell \mid \phi_\ell \wedge \neg(\exists \vec{x}.\ (t_\ell = t_r) \wedge \phi_r)\rangle \Rightarrow \langle t_r \mid \phi_r\rangle \qquad (t_\ell = t_r) \wedge \phi_r \text{ is satisfiable}}{\langle t_\ell \mid \phi_\ell\rangle \Rightarrow \langle t_r \mid \phi_r\rangle}\ (\mathsf{subs})$$

where $\mathcal{V}ar(t_r,\phi_r) \setminus \mathcal{V}ar(t_\ell,\phi_\ell) = \{\vec{x}\}$. A term $t_\ell\gamma_\ell \in [\![\langle t_\ell \mid \phi_\ell\rangle]\!]$ with γ_ℓ respecting ϕ_ℓ is in $[\![\langle t_r \mid \phi_r\rangle]\!]$ if there is some substitution γ_r such that $t_r\gamma_r = t_\ell\gamma_\ell$ and γ_r respects ϕ_r, i.e., t_ℓ and t_r are unifiable and their unifier respects $\phi_\ell \wedge \phi_r$. To remove such a term from $[\![\langle t_\ell \mid \phi_\ell\rangle]\!]$, the formula $\neg(\exists \vec{x}.\ (t_\ell = t_r) \wedge \phi_r)$ is conjunct to ϕ_ℓ.

Since this paper uses the original definition of LCTRSs [10], we cannot use DCC as it is. Our approach to the simplification is mainly to restrict APR problems to some class. On the other hand, the restriction makes some side conditions of proof rules hold, and we simplify the formulation of the proof system.

APR problems for runtime-error verification handled in this paper have the singleton set $\langle \mathsf{success} \mid \mathsf{true}\rangle$ of destinations, and success is a normal form. Therefore, there is no need to consider constraints of structural equivalence such as $t_\ell = t_r$ in subs: The application of subs relies on $t_\ell = \mathsf{success}$; if $t_\ell = \mathsf{success}$, then the demonically valid problem $\langle t_\ell \mid \phi_\ell \wedge \mathsf{false}\rangle \Rightarrow \langle t_r \mid \phi_r\rangle$ is inferred and proved by axiom; otherwise, the application of the rule is redundant because $[\![\langle t_\ell \mid \phi_\ell \wedge \neg(\exists \vec{x}.\ t_\ell = t_r \wedge \phi_r)\rangle]\!] = [\![\langle t_\ell \mid \phi_\ell\rangle]\!]$. Thus, "$t_\ell = \mathsf{success}$" is enough for APR problems handled in this paper and we do not have to consider structural equivalence in the side condition of rules in DCC.

From the above observation, we define a class of APR problems whose destination is a constant.

Definition 5.1 (constant-directed APR problem) *An APR problem $\langle t_\ell \mid \phi_\ell\rangle \Rightarrow \langle t_r \mid \phi_r\rangle$ is called* constant-directed *if t_r is a constant normal form and ϕ_r is satisfiable.*

In the following, w.l.o.g., we deal with constant-directed problems of the form $\langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle$.[5]

The proof system DSTEP in [4] is defined for constant-directed problems as follows.

Definition 5.2 (a simplified version of DSTEP) *Given an LCTRS $\mathcal{R}$, the proof system* $\mathsf{DSTEP}(\mathcal{R})$ *consists of the following proof rules:*

$$\frac{\phi_\ell \text{ is unsatisfiable}}{\langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle}\ (\mathsf{axiom}) \qquad \frac{t_\ell = c}{\langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle}\ (\mathsf{subs})$$

$$\frac{\langle t_1 \mid \phi_1 \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle \ldots \langle t_n \mid \phi_n \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle \quad \langle t_\ell \mid \phi_\ell \rangle \text{ is } \mathcal{R}\text{-derivable} \quad [\![\langle t_\ell \mid \phi_\ell \rangle]\!] \cap NF_{\mathcal{R}} = \emptyset}{\langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle}\ (\mathsf{der})^{6}$$

where $\Delta_{\mathcal{R}}(\langle t_\ell \mid \phi_\ell \rangle) = \{\langle t_i \mid \phi_i \rangle \mid 1 \leq i \leq n\}$ *for some* $n > 0$.

Note that $\langle t_\ell \mid \phi_\ell \rangle$ is $\mathcal{R}$-derivable and $[\![\langle t_\ell \mid \phi_\ell \rangle]\!] \cap NF_{\mathcal{R}} = \emptyset$ if and only if $[\![\langle t_\ell \mid \phi_\ell \rangle]\!]$ is runnable w.r.t. $\to_{\mathcal{R}}$. For our problem $\langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true} \rangle$ of the LCTRS $\mathcal{R}^\forall$, t_ℓ has sort *cnfg*, and by the construction of $\mathcal{R}^\forall$, every term $\mathsf{cnfg}(\ldots)$ with sort *cnfg* can be reduced to success.[7] For this reason, in our setting, the side condition $[\![\langle t_\ell \mid \phi_\ell \rangle]\!] \cap NF_{\mathcal{R}} = \emptyset$ always holds.

Theorem 5.3 *Let $\mathcal{R}$ be an LCTRS. For a constant-directed APR problem $\langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle$ of $\mathcal{R}$, $\mathcal{R} \models^\forall \langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle$ if and only if $\langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle \in \nu\widehat{\mathsf{DSTEP}}$.*[8]

Proof (Sketch). This theorem holds for DSTEP in [4]. The rules in Definition 5.2 are just reformulations of those in [4] for constant-directed problems. Therefore, this theorem holds. □

Given a finite set G of constant-directed APR problems, let us consider the following rule circ of DCC for *circularity*, which is reformulated for constant-directed problems:

$$\frac{\langle c' \mid \phi_\ell \wedge (\exists \vec{x}.\ (t_\ell = t'_\ell) \wedge \phi'_\ell) \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle \qquad \langle t_\ell \mid \phi_\ell \wedge \neg(\exists \vec{x}.\ (t_\ell = t'_\ell) \wedge \phi'_\ell) \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle}{\langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle}\ (\mathsf{circ})$$

where $\{\vec{x}\} = \mathcal{V}ar(t'_\ell, \phi'_\ell)$ and $\langle t'_\ell \mid \phi'_\ell \rangle \Rightarrow \langle c' \mid \mathsf{true} \rangle$ is a freshly renamed problem in G. The problem $\langle t'_\ell \mid \phi'_\ell \rangle \Rightarrow \langle c' \mid \mathsf{true} \rangle$ does not have to be constant-directed, and may perform as a lemma to prove a main problem. On the other hand, it is difficult to implement a function to automatically find such a lemma. For this reason, for the present, we deal with constant-directed problems only. Constants with sort *cnfg* in our APR problems are unique, i.e., success, and thus, we assume that $c' = c$ (= success). Then, $\langle c' \mid \phi_\ell \wedge (\exists \vec{x}.\ (t_\ell = t'_\ell) \wedge \phi'_\ell) \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle$ can be proved by subs, reformulating circ as follows:

$$\frac{\langle t_\ell \mid \phi_\ell \wedge \neg(\exists \vec{x}.\ (t_\ell = t'_\ell) \wedge \phi'_\ell) \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle}{\langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle c \mid \mathsf{true} \rangle}\ (\mathsf{circ})$$

The above reformulation still contains structural-equivalence formula $t_\ell = t'_\ell$. To drop the formula, we focus on the following property.

[5] It is clear that a constant-directed problem $\langle t_\ell \mid \phi_\ell \rangle \Rightarrow \langle c \mid \phi_r \rangle$ is equivalent to the constant-directed problem $\langle t_\ell \mid \phi_\ell \wedge (\exists \vec{x}.\ \phi_r) \rangle \Rightarrow \langle t \mid \mathsf{true} \rangle$, where $\{\vec{x}\} = \mathcal{V}ar(\phi_r) \setminus \mathcal{V}ar(t_\ell, \phi_\ell)$.

[6] The original definition of der uses "validity of $\phi_\ell \to \bigvee_{j \in \{1,\ldots,n\}} \exists \vec{y_j}.\ \phi_j$" where $\{\vec{y_j}\} = \mathcal{V}ar(t_j, \phi_j) \setminus \mathcal{V}ar(t_\ell, \phi_\ell)$. This condition implies that for any instance of $\langle t_\ell \mid \phi_\ell \rangle$, there exists at least one rewrite rule that is applied to the instance [4]. For this reason, the condition is equivalent to $[\![\langle t_\ell \mid \phi_\ell \rangle]\!] \cap NF_{\mathcal{R}} = \emptyset$.

[7] It is clear that if ϕ_ℓ is satisfiable and t_ℓ is rooted by cnfg (i.e., $t_\ell \notin \{\mathsf{success}, \mathsf{error}\}$) if and only if $\langle t_\ell \mid \phi_\ell \rangle$ is $\mathcal{R}^\forall$-derivable.

[8] For a set *RULES* of rules, $\nu\widehat{RULES}$ stands for the greatest fixed point of $\widehat{RULES}$ which stands for the functional of *RULES*.

Proposition 5.4 *Let $\langle s \mid \phi\rangle, \langle t \mid \psi\rangle$ be constrained terms, and $\{\vec{x}\} = \mathcal{V}ar(t,\psi)$ such that $\mathcal{V}ar(s,\phi) \cap \{\vec{x}\} = \emptyset$. If $[\![\langle s \mid \phi\rangle]\!] \subseteq [\![\langle t \mid \psi\rangle]\!]$, then $\phi \wedge \neg(\exists\vec{x}.\ (s = t) \wedge \psi)$ is unsatisfiable.*

By the above proposition, if $[\![\langle t_\ell \mid \phi_\ell\rangle]\!] \subseteq [\![\langle t'_\ell \mid \phi'_\ell\rangle]\!]$, then the problem $\langle t_\ell \mid \phi_\ell \wedge \neg(\exists\vec{x}.\ (t_\ell = t'_\ell) \wedge \phi'_\ell)\rangle \Rightarrow \langle c \mid \mathsf{true}\rangle$ can be proved by axiom. By forcing $[\![\langle t_\ell \mid \phi_\ell\rangle]\!] \subseteq [\![\langle t'_\ell \mid \phi'_\ell\rangle]\!]$ as a side condition, we remove the APR problem from the premise of circ. To distinguish such a new rule with the original one, we call it weak circ.

Definition 5.5 (a proof system DCC^-) *Let $\mathcal{R}$ be an LCTRS, and G a finite set of constant-directed APR problems. Then, the proof system $\mathsf{DCC}^-(\mathcal{R},G)$ consists of $\mathsf{DSTEP}(\mathcal{R})$, together with*

$$\frac{\exists\langle t'_\ell \mid \phi'_\ell\rangle \Rightarrow \langle c \mid \mathsf{true}\rangle \in G \text{ such that } [\![\langle t_\ell \mid \phi_\ell\rangle]\!] \subseteq [\![\langle t'_\ell \mid \phi'_\ell\rangle]\!]}{\langle t_\ell \mid \phi_\ell\rangle \Rightarrow \langle c \mid \mathsf{true}\rangle}\ (\mathsf{weak\ circ})$$

Since DCC^- is a weaker reformulation of DCC, the soundness proof [4, Theorem 3] for DCC ensures soundness of DCC^-.

Theorem 5.6 (soundness of DCC^-) *Let $\mathcal{R}$ be an LCTRS, and G a finite set of constant-directed APR problems. Suppose that for each problem $\langle t_\ell \mid \phi_\ell\rangle \Rightarrow \langle c \mid \mathsf{true}\rangle \in G$, there exists a proof tree $\mathcal{T}$ under $\mathsf{DCC}^-(\mathcal{R},G)$, each circ node of which has a der node as an ancestor. Then, $\mathcal{R} \models^{\forall} \langle t_\ell \mid \phi_\ell\rangle \Rightarrow \langle c \mid \mathsf{true}\rangle$ for all problems $\langle t_\ell \mid \phi_\ell\rangle \Rightarrow \langle c \mid \mathsf{true}\rangle \in G$.*

In proving an APR problem, as for *cyclic proofs* [2], it is enough to construct a single proof tree under $\mathsf{DCC}(\mathcal{R},G)$ or $\mathsf{DCC}^-(\mathcal{R},G)$ such that

- G includes the APR problem and all der nodes in the tree,
- the root node is the APR problem, and
- for each (weak) circ node, the tree includes a der node, the conclusion of which is the same as the (weak) circ node.

Example 5.7 Consider the LCTRS $\mathcal{R}_1^{\forall}$ in Fig. 3 and the following APR problem in Example 4.2 again:

$$\langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_7,0),\mathsf{p}(\mathsf{rdr}_7,0),\mathsf{p}(\mathsf{wtr}_{18},0),\mathsf{sem}(2,1,3),0,0) \mid \mathsf{true}\rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true}\rangle$$

Since the second and third arguments of sem are incremented during the reduction, we cannot rely on circularity in proving the demonical validity of the above problem. For this reason, we prove the demonical validity of the following generalized APR problem:

$$(1)\ \langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_7,0),\mathsf{p}(\mathsf{rdr}_7,0),\mathsf{p}(\mathsf{wtr}_{18},0),\mathsf{sem}(2,d,t),x,0) \mid d > 0 \wedge t = d+2\rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true}\rangle$$

Since the size of proof trees for the above problem is very large, we show how to infer sub problems in applying DCC^- rules to some problems obtained from the above generalized problem. By applying der to the above generalized problem, we obtain the following APR problems:

$$(1.1)\ \langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_9(y_1),0),\mathsf{p}(\mathsf{rdr}_7,0),\mathsf{p}(\mathsf{wtr}_{18},0),\mathsf{sem}(2,d,t),x,0) \mid d > 0 \wedge t = d+2\rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true}\rangle$$
$$(1.2)\ \langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_7,0),\mathsf{p}(\mathsf{rdr}_9(y_2),0),\mathsf{p}(\mathsf{wtr}_{18},0),\mathsf{sem}(2,d,t),x,0) \mid d > 2 \wedge t = d+2\rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true}\rangle$$
$$(1.3)\ \langle \mathsf{cnfg}(\mathsf{p}(\mathsf{rdr}_7,0),\mathsf{p}(\mathsf{rdr}_7,0),\mathsf{p}(\mathsf{wtr}_{19},0),\mathsf{sem}(1,d,t),x,0) \mid d > 0 \wedge t = d+2\rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true}\rangle$$
$$(1.4)\ \langle \mathsf{cnfg}(\ldots,\mathsf{p}(\mathsf{wtr}_{18},t),\mathsf{sem}(1,d,t'),x,0) \mid d > 0 \wedge t = d+2 \wedge 2 = 0 \wedge t' = t+2\rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true}\rangle$$
$$(1.5)\ \langle \mathsf{cnfg}(\ldots,\mathsf{p}(\mathsf{wtr}_{19},0),\mathsf{sem}(1,d,t),x,0) \mid 0 = d \wedge 0 \neq 0 \wedge d > 0 \wedge t = d+2\rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true}\rangle$$
$$(1.6)\ \langle \mathsf{success} \mid \mathsf{true}\rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true}\rangle$$

By applying axiom, (1.4) and (1.5) are proved. By applying subs, (1.6) is proved. We apply der to (1.1)–(1.3) again. Let us focus on (1.1). Repeating the application of der, we obtain (1) again. The second occurrence of (1) can be proved by weak circ with the first occurrence of (1). In this way, we can construct a proof tree for (1) under $\mathsf{DCC}^{-}(\mathcal{R}_1^{\forall}, \{(1)\})$. Therefore, Theorem 5.6 ensures the race-freeness of exclusive control in Program 1.

Finally, we show a sufficient condition for $[\![\langle t_\ell \mid \phi_\ell \rangle]\!] \subseteq [\![\langle t'_\ell \mid \phi'_\ell \rangle]\!]$ in weak circ.

Proposition 5.8 *Let $\langle s \mid \phi \rangle, \langle t \mid \psi \rangle$ be constrained terms, and $\{\vec{x}\} = \mathcal{V}ar(t, \psi)$ such that $\mathcal{V}ar(s, \phi) \cap \{\vec{x}\} = \emptyset$. If there exists a substitution γ such that $\mathcal{R}an(\gamma|_{\mathcal{V}ar(\psi)}) \subseteq T(\Sigma_{theory}, \mathcal{V}ar(\phi))$, $s = t\gamma$, and $\phi \iff \psi\gamma$ is valid, then $[\![\langle s \mid \phi \rangle]\!] \subseteq [\![\langle t \mid \psi \rangle]\!]$.*

6 Conclusion

In this paper, we proposed a framework to reduce the non-occurrence of a specified runtime error to an APR problem and also proposed a simplified variant of the proof system DCC. We have implemented DCC^{-} in Crisys2, a tool based on RI of LCTRSs, by restricting and relaxing the side conditions of inference rules for RI to those for DCC^{-}. The tool succeeded in automatically proving the APR problems shown in this paper.

The examples in this paper are very simple, e.g., the simple reader-writer problem. Our future work is to apply the framework in this paper to more practical programs such as in-vehicle embedded systems. To this end, it would be necessary to increase the functions of concurrent programs that can be handled, such as interruption processing, and we will apply the framework to verification of various runtime errors. Furthermore, we will compare our method with model checking regarding the non-occurrence of runtime errors.

Acknowledgements We gratefully acknowledge the anonymous reviewers for their useful comments and suggestions to improve the paper.

References

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

[2] James Brotherston (2005): *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*, *Lecture Notes in Computer Science* 3702, Springer, pp. 78–92, doi:10.1007/11554554_8.

[3] Andrei-Sebastian Buruiană & Ştefan Ciobâcă (2018): *Reducing Total Correctness to Partial Correctness by a Transformation of the Language Semantics*. In Joachim Niehren & David Sabel, editors: *Proceedings of the 5th International Workshop on Rewriting Techniques for Program Transformations and Evaluation*, *Electronic Proceedings in Theoretical Computer Science* 289, Open Publishing Association, pp. 1–16, doi:10.4204/EPTCS.289.1.

[4] Ştefan Ciobâcă & Dorel Lucanu (2018): *A Coinductive Approach to Proving Reachability Properties in Logically Constrained Term Rewriting Systems*. In Didier Galmiche, Stephan Schulz & Roberto Sebastiani, editors: *Proceedings of the 9th International Joint Conference on Automated Reasoning*, *Lecture Notes in Computer Science* 10900, Springer, pp. 295–311, doi:10.1007/978-3-319-94205-6_20.

[5] Carsten Fuhs, Cynthia Kop & Naoki Nishida (2017): *Verifying Procedural Programs via Constrained Rewriting Induction*. *ACM Transactions on Computational Logic* 18(2), pp. 14:1–14:50, doi:10.1145/3060143.

[6] Yoshiaki Kanazawa & Naoki Nishida (2019): *On Transforming Functions Accessing Global Variables into Logically Constrained Term Rewriting Systems*. In Joachim Niehren & David Sabel, editors: *Proceedings of the 5th International Workshop on Rewriting Techniques for Program Transformations and Evaluation*, *Electronic Proceedings in Theoretical Computer Science* 289, Open Publishing Association, pp. 34–52.

[7] Yoshiaki Kanazawa, Naoki Nishida & Masahiko Sakai (2019): *On Representation of Structures and Unions in Logically Constrained Rewriting*. IEICE Technical Report SS2018-38, IEICE. Vol. 118, No. 385, pp. 67–72, in Japanese.

[8] Misaki Kojima (2022): *Runtime-Error Verification of Programs with Exclusive Control by Reducing to All-Path Reachability in Constrained Rewriting*. Master's thesis, Graduate School of Informatics, Nagoya University. In Japanese.

[9] Misaki Kojima, Naoki Nishida & Yutaka Matsubara (2020): *Transforming Concurrent Programs with Semaphores into Logically Constrained Term Rewrite Systems*. In: *Informal Proceedings of the 7th International Workshop on Rewriting Techniques for Program Transformations and Evaluation*, pp. 1–12.

[10] Cynthia Kop & Naoki Nishida (2013): *Term Rewriting with Logical Constraints*. In Pascal Fontaine, Christophe Ringeissen & Renate A. Schmidt, editors: *Proceedings of the 9th International Symposium on Frontiers of Combining Systems*, *Lecture Notes in Computer Science* 8152, Springer, pp. 343–358, doi:10.1007/978-3-642-40885-4_24.

[11] Naoki Nishida & Sarah Winkler (2018): *Loop Detection by Logically Constrained Term Rewriting*. In Ruzica Piskac & Philipp Rümmer, editors: *Proceedings of the 10th Working Conference on Verified Software: Theories, Tools, and Experiments*, *Lecture Notes in Computer Science* 11294, Springer, pp. 309–321, doi:10.1007/978-3-030-03592-1_18.

[12] Enno Ohlebusch (2002): *Advanced Topics in Term Rewriting*. Springer, doi:10.1007/978-1-4757-3661-8.

[13] Uday S. Reddy (1990): *Term Rewriting Induction*. In Mark E. Stickel, editor: *Proceedings of the 10th International Conference on Automated Deduction*, *Lecture Notes in Computer Science* 449, Springer, pp. 162–177, doi:10.1007/3-540-52885-7_86.

[14] Andrei Stefanescu, Ştefan Ciobâcă, Radu Mereuta, Brandon M. Moore, Traian-Florin Serbanuta & Grigore Rosu (2014): *All-Path Reachability Logic*. In Gilles Dowek, editor: *Proceedings of the Joint International Conference on Rewriting and Typed Lambda Calculi*, *Lecture Notes in Computer Science* 8560, Springer, pp. 425–440, doi:10.1007/978-3-319-08918-8_29.

[15] Andrei Stefanescu, Ştefan Ciobâcă, Radu Mereuta, Brandon M. Moore, Traian-Florin Serbanuta & Grigore Rosu (2019): *All-Path Reachability Logic*. *Logical Methods in Computer Science* 15(2), doi:10.23638/LMCS-15(2:5)2019.

[16] Andrew S. Tanenbaum (2001): *Modern operating systems, 2nd Edition*. Prentice Hall.

[17] Andrew S. Tanenbaum & Albert S. Woodhull (2006): *Operating systems — design and implementation*, 3 edition. Pearson Education.

[18] Sarah Winkler & Aart Middeldorp (2018): *Completion for Logically Constrained Rewriting*. In Hélène Kirchner, editor: *Proceedings of the 3rd International Conference on Formal Structures for Computation and Deduction*, *LIPIcs* 108, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, pp. 30:1–30:18, doi:10.4230/LIPIcs.FSCD.2018.30.

A Missing Proofs

Theorem 4.1 *(1)* $\langle s_0 \mid \phi_0 \rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true} \rangle$ *is demonically valid w.r.t.* $\mathcal{R}^{\forall}$ *if and only if (2)* $s \not\to^*_{\mathcal{R}} u$ *for any ground term* $s \in [\![\langle s_0 \mid \phi_0 \rangle]\!]$ *and any ground term* $u \in \bigcup_{i=1}^{n} [\![\langle u_i \mid \phi_i \rangle]\!]$.

Proof. We first show the *only-if* part. Assume that (1) holds but (2) does not. Then, there exist ground terms $s \in [\![\langle s_0 \mid \phi_0 \rangle]\!]$ and $u \in \bigcup_{i=1}^{n} [\![\langle u_i \mid \phi_i \rangle]\!]$ such that $s \to_{\mathcal{R}}^{*} u$. By the construction of $\mathcal{R}^{\forall}$, it is clear that $\mathcal{R} \subseteq \mathcal{R}^{\forall}$ and $u \to_{\mathcal{R}^{\forall}} \mathsf{error}$, and hence $s \to_{\mathcal{R}^{\forall}}^{*} u \to_{\mathcal{R}^{\forall}} \mathsf{error}$. Since $\mathsf{success}$ is a normal form of $\mathcal{R}^{\forall}$, $\mathsf{success}$ does not occur in any rewrite sequence of $s \to_{\mathcal{R}^{\forall}}^{*} u \to_{\mathcal{R}^{\forall}} \mathsf{error}$. This implies that (1) does not holds, contradicting the assumption that (1) holds.

Next, we show the *if* part. Assume that (2) holds but (1) does not. Then, there exists a finite execution path that starts with $s \in [\![\langle s_0 \mid \phi_0 \rangle]\!]$ and ends with error, i.e., $s \to_{\mathcal{R}^{\forall}}^{*} \mathsf{error}$. It follows from the construction of $\mathcal{R}^{\forall}$ that there exist a natural number $i \in \{1, \ldots, n\}$ and a ground term $u \in [\![\langle u_i \mid \phi_i \rangle]\!]$ such that $s \to_{\mathcal{R}^{\forall}}^{*} u \to_{\mathcal{R}^{\forall}} \mathsf{error}$. Neither normal forms $\mathsf{success}$ nor error of $\mathcal{R}^{\forall}$ occurs in any rewrite sequence for $s \to_{\mathcal{R}^{\forall}}^{*} u$, i.e., $s \to_{\mathcal{R}}^{*} u$. This contradicts the assumption that (2) holds. □

Proposition 5.4 *Let $\langle s \mid \phi \rangle, \langle t \mid \psi \rangle$ be constrained terms, and $\{\vec{x}\} = \mathcal{V}ar(t, \psi)$ such that $\mathcal{V}ar(s, \phi) \cap \{\vec{x}\} = \emptyset$. If $[\![\langle s \mid \phi \rangle]\!] \subseteq [\![\langle t \mid \psi \rangle]\!]$, then $\phi \wedge \neg(\exists \vec{x}.\ (s = t) \wedge \psi)$ is unsatisfiable.*

Proof. We proceed by contradiction. Assume that $[\![\langle s \mid \phi \rangle]\!] \subseteq [\![\langle t \mid \psi \rangle]\!]$ and $\phi \wedge \neg(\exists \vec{x}.\ (s = t) \wedge \psi)$ is satisfiable. Let γ be a substitution such that (a) $\mathcal{D}om(\gamma) = \mathcal{V}ar(s, \phi)$, (b) $[\![\phi\gamma]\!] = \top$, (c) $[\![\neg(\exists \vec{x}.\ (s\gamma = t) \wedge \psi)]\!] = \top$. Then, γ respects ϕ, and hence $s\gamma \in [\![\langle s \mid \phi \rangle]\!]$. Let γ' be a substitution such that $\mathcal{D}om(\gamma') = \mathcal{V}ar(s, \phi, t, \psi)$ and $\gamma'|_{\mathcal{V}ar(s,\phi)} = \gamma$. We make a case analysis depending on whether $s\gamma' = t\gamma'$ holds or not.

- Consider the case where $s\gamma' = t\gamma'$. It follows from (c) that $[\![\psi\gamma']\!] = \bot$, and hence $t\gamma' \notin [\![\langle t \mid \psi \rangle]\!]$.
- Consider the remaining case where $s\gamma' \neq t\gamma'$. By definition, it is clear that $t\gamma' \notin [\![\langle t \mid \psi \rangle]\!]$.

In both cases above, we have that $t\gamma' \notin [\![\langle t \mid \psi \rangle]\!]$, and hence $s\gamma \notin [\![\langle t \mid \psi \rangle]\!]$. This contradicts the assumption that $[\![\langle s \mid \phi \rangle]\!] \subseteq [\![\langle t \mid \psi \rangle]\!]$. □

Proposition 5.8 *Let $\langle s \mid \phi \rangle, \langle t \mid \psi \rangle$ be constrained terms, and $\{\vec{x}\} = \mathcal{V}ar(t, \psi)$ such that $\mathcal{V}ar(s, \phi) \cap \{\vec{x}\} = \emptyset$. If there exists a substitution γ such that $\mathcal{R}an(\gamma|_{\mathcal{V}ar(\psi)}) \subseteq T(\Sigma_{theory}, \mathcal{V}ar(\phi))$, $s = t\gamma$, and $\phi \iff \psi\gamma$ are valid, then $[\![\langle s \mid \phi \rangle]\!] \subseteq [\![\langle t \mid \psi \rangle]\!]$.*

Proof. Let s' be a term in $[\![\langle s \mid \phi \rangle]\!]$. Then, there exists a substitution θ such that $\mathcal{D}om(\theta) = \mathcal{V}ar(s, \phi)$ and θ respects ϕ, i.e., $\mathcal{R}an(\theta|_{\mathcal{V}ar(\phi)}) \subseteq \mathcal{V}al$ and $[\![\phi\theta]\!] = \top$. Then, there exists a substitution γ' such that $\mathcal{D}om(\gamma') = \mathcal{V}ar(t\gamma, \psi\gamma)$, $s\theta = t\gamma\gamma'$, and $\mathcal{R}an(\gamma'|_{\mathcal{V}ar(\psi\gamma)} \subseteq \mathcal{V}al$. It follows from the validity of $\phi \iff \psi\gamma$ that $[\![\psi\gamma\gamma']\!] = \top$, and hence, γ' respects $\psi\gamma$. Therefore, $t\gamma\gamma' \in [\![t \mid \psi]\!]$, i.e., $s\theta \in [\![t \mid \psi]\!]$. □

B Example: Sleeping Barber Problem

Program 2 is a pseudo code for the *sleeping barber problem* with `N` seats [16]. In the following, We let $\mathtt{N} = 2$. Let us consider a configuration with two processes, the first of which executes `barber` in Program 2, and the second of which executes `customer` in Program 2. Program 2 is transformed into the LCTRS $\mathcal{R}_2 = \mathcal{R}_{\mathsf{bbr}} \cup \mathcal{R}_{\mathsf{cstmr}}$ in Fig. 4 and 5, together with the following initial term:

$$\mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_8,0,0,0), \mathsf{p}(\mathsf{cstmr}_{18},0,0,0), \mathsf{sem}(0,1,3), \mathsf{sem}(0,1,3), \mathsf{sem}(1,1,3), 2, 0)$$

Program 2: a program of the sleeping barber problem with `N` seats

```
1  semaphore Customers = 0;
2  semaphore Barber = 0;
3  semaphore accessSeats = 1;
4  int NumberOfFreeSeats = N;
5
6  void barber(void){
7    while(true){
8      down(Customers);
9      down(accessSeats);
10     NumberOfFreeSeats ++;
11     up(Barber);
12     up(accessSeats);
13   }
14 }
```

```
15
16 void customer(void){
17   while(true){
18     down(accessSeats);
19     if(NumberOfFreeSeats > 0){
20       NumberOfFreeSeats --;
21       up(Customers);
22       up(accessSeats);
23       down(Barber);
24     }else{
25       up(accessSeats);
26     }
27   }
28 }
```

For $\mathcal{R}_2$ and the initial term, the following LCTRS and APR problem for the race-freeness of all the semaphores in Program 2 are generated:

$$\mathcal{R}_2^{\forall} = \mathcal{R}_{\mathsf{bbr}} \cup \mathcal{R}_{\mathsf{cstmr}} \cup \left\{ \begin{array}{r} \mathsf{cnfg}(p_1, p_2, s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{success} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{10}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{19}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{11}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{19}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{12}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{19}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{10}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{20}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{11}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{20}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{12}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{20}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{10}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{21}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{11}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{21}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{12}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{21}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{10}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{22}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{11}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{22}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{12}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{22}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{10}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{25}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{11}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{25}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \\ \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{12}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{p}(\mathsf{cstmr}_{25}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, f) \to \mathsf{error} \end{array} \right\}$$

and

$$\langle \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_8,0,0,0), \mathsf{p}(\mathsf{cstmr}_{18},0,0,0), \mathsf{sem}(0,1,3), \mathsf{sem}(0,1,3), \mathsf{sem}(1,1,3), 2, 0) \mid \mathsf{true} \rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true} \rangle$$

As for Example 5.7, the generalization of the above APR problem is necessary for the use of circularity:

$$\langle \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_8,0,0,0), \mathsf{p}(\mathsf{cstmr}_{18},0,0,0), \mathsf{sem}(0,d_b,t_b), \mathsf{sem}(0,d_c,t_c), \mathsf{sem}(1,d_s,t_s), 2, 0) \mid \phi_2 \rangle \Rightarrow \langle \mathsf{success} \mid \mathsf{true} \rangle$$

where $\phi_2 = d_b > 0 \wedge t_b = d_b + 2 \wedge d_c > 0 \wedge t_c = d_c + 2 \wedge d_s > 0 \wedge t_s = d_s + 2$. The proof system DCC^- succeeds in proving the above APR problem.

$$
\mathcal{R}_{\mathsf{bbr}} = \left\{
\begin{array}{l}
\mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_8, n_{\mathrm{bar}}, 0, n_{\mathrm{seat}}), p_{\mathrm{cus}}, s_{\mathrm{bar}}, \mathsf{sem}(s_{\mathrm{cus}}, d_{\mathrm{cus}}, t_{\mathrm{cus}}), s_{\mathrm{seat}}, \mathit{frst}, 0) \\
\quad \to \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_9, n_{\mathrm{bar}}, 0, n_{\mathrm{seat}}), p_{\mathrm{cus}}, s_{\mathrm{bar}}, \mathsf{sem}(s_{\mathrm{cus}} - 1, d_{\mathrm{cus}}, t_{\mathrm{cus}}), s_{\mathrm{seat}}, \mathit{frst}, 0) \quad [s_{\mathrm{cus}} \neq 0] \\
\mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_8, n_{\mathrm{bar}}, 0, n_{\mathrm{seat}}), p_{\mathrm{cus}}, s_{\mathrm{bar}}, \mathsf{sem}(s_{\mathrm{cus}}, d_{\mathrm{cus}}, t_{\mathrm{cus}}), s_{\mathrm{seat}}, \mathit{frst}, 0) \\
\quad \to \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_8, n_{\mathrm{bar}}, t_{\mathrm{cus}}, n_{\mathrm{seat}}), p_{\mathrm{cus}}, s_{\mathrm{bar}}, \mathsf{sem}(s_{\mathrm{cus}}, d_{\mathrm{cus}}, t_{\mathrm{cus}} + 2), s_{\mathrm{seat}}, \mathit{frst}, 0) \quad [s_{\mathrm{cus}} = 0] \\
\mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_8, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), p_{\mathrm{cus}}, s_{\mathrm{bar}}, \mathsf{sem}(s_{\mathrm{cus}}, d_{\mathrm{cus}}, t_{\mathrm{cus}}), s_{\mathrm{seat}}, \mathit{frst}, 1) \\
\quad \to \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_9, n_{\mathrm{bar}}, 0, n_{\mathrm{seat}}), p_{\mathrm{cus}}, s_{\mathrm{bar}}, \mathsf{sem}(s_{\mathrm{cus}}, d_{\mathrm{cus}}, t_{\mathrm{cus}}), s_{\mathrm{seat}}, \mathit{frst}, 0) \quad [n_{\mathrm{cus}} = d_{\mathrm{cus}} \wedge n_{\mathrm{cus}} \neq 0] \\
\\
\mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_9, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), p_{\mathrm{cus}}, s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) \\
\quad \to \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{10}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), p_{\mathrm{cus}}, s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}} - 1, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) \quad [s_{\mathrm{seat}} \neq 0] \\
\mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_9, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), p_{\mathrm{cus}}, s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) \\
\quad \to \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_9, n_{\mathrm{bar}}, n_{\mathrm{cus}}, t_{\mathrm{seat}}), p_{\mathrm{cus}}, s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}} + 2), \mathit{frst}, 0) \, [s_{\mathrm{seat}} = 0] \\
\mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_9, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), p_{\mathrm{cus}}, s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 1) \\
\quad \to \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{10}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), p_{\mathrm{cus}}, s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) \quad [n_{\mathrm{seat}} = d_{\mathrm{seat}} \wedge n_{\mathrm{seat}} \neq 0] \\
\\
\mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{10}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), p_{\mathrm{cus}}, s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) \\
\quad \to \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{11}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), p_{\mathrm{cus}}, s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst} + 1, 0) \\
\\
\mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{11}, 0, n_{\mathrm{cus}}, n_{\mathrm{seat}}), p_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{bar}}, d_{\mathrm{bar}}, t_{\mathrm{bar}}), s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) \\
\quad \to \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{12}, 0, n_{\mathrm{cus}}, n_{\mathrm{seat}}), p_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{bar}}, d_{\mathrm{bar}} + 2, t_{\mathrm{bar}}), s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 1) \quad [t_{\mathrm{bar}} \neq d_{\mathrm{bar}} + 2] \\
\mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{11}, 0, n_{\mathrm{cus}}, n_{\mathrm{seat}}), p_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{bar}}, d_{\mathrm{bar}}, t_{\mathrm{bar}}), s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) \\
\quad \to \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{12}, 0, n_{\mathrm{cus}}, n_{\mathrm{seat}}), p_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{bar}} + 1, d_{\mathrm{bar}}, t_{\mathrm{bar}}), s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) \quad [t_{\mathrm{bar}} = d_{\mathrm{bar}} + 2] \\
\\
\mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{12}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), p_{\mathrm{cus}}, s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) \\
\quad \to \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_8, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), p_{\mathrm{cus}}, s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}} + 2, t_{\mathrm{seat}}), \mathit{frst}, 1) \quad [t_{\mathrm{seat}} \neq d_{\mathrm{seat}} + 2] \\
\mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_{12}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), p_{\mathrm{cus}}, s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) \\
\quad \to \mathsf{cnfg}(\mathsf{p}(\mathsf{bbr}_8, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), p_{\mathrm{cus}}, s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}} + 1, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) \quad [t_{\mathrm{seat}} = d_{\mathrm{seat}} + 2]
\end{array}
\right\}
$$

Figure 4: LCTRS $\mathcal{R}_{\mathsf{bbr}}$ representing the first process executing `barber` in Program 2

$$
\mathcal{R}_{\mathsf{cstmr}} = \left\{
\begin{array}{ll}
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{18}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{19}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}} - 1, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) & [s_{\mathrm{seat}} \neq 0] \\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{18}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{18}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, t_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}} + 2), \mathit{frst}, 0) & [s_{\mathrm{seat}} = 0] \\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{18}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 1) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{19}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) & [n_{\mathrm{seat}} = d_{\mathrm{seat}} \wedge n_{\mathrm{seat}} \neq 0] \\
\\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{19}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{20}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) & [\mathit{frst} > 0] \\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{19}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{25}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) & [\mathit{frst} \leq 0] \\
\\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{20}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{21}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), s_{\mathrm{bar}}, s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst} - 1, 0) & \\
\\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{21}, n_{\mathrm{bar}}, 0, n_{\mathrm{seat}}), s_{\mathrm{bar}}, \mathsf{sem}(s_{\mathrm{cus}}, d_{\mathrm{cus}}, t_{\mathrm{cus}}), s_{\mathrm{seat}}, \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{22}, n_{\mathrm{bar}}, 0, n_{\mathrm{seat}}), s_{\mathrm{bar}}, \mathsf{sem}(s_{\mathrm{cus}}, d_{\mathrm{cus}} + 2, t_{\mathrm{cus}}), s_{\mathrm{seat}}, \mathit{frst}, 1) & [t_{\mathrm{cus}} \neq d_{\mathrm{cus}} + 2] \\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{21}, n_{\mathrm{bar}}, 0, n_{\mathrm{seat}}), s_{\mathrm{bar}}, \mathsf{sem}(s_{\mathrm{cus}}, d_{\mathrm{cus}}, t_{\mathrm{cus}}), s_{\mathrm{seat}}, \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{22}, n_{\mathrm{bar}}, 0, n_{\mathrm{seat}}), s_{\mathrm{bar}}, \mathsf{sem}(s_{\mathrm{cus}} + 1, d_{\mathrm{cus}}, t_{\mathrm{cus}}), s_{\mathrm{seat}}, \mathit{frst}, 0) & [t_{\mathrm{cus}} = d_{\mathrm{cus}} + 2] \\
\\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{22}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{23}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}} + 2, t_{\mathrm{seat}}), \mathit{frst}, 1) & [t_{\mathrm{seat}} \neq d_{\mathrm{seat}} + 2] \\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{22}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{23}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}} + 1, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) & [t_{\mathrm{seat}} = d_{\mathrm{seat}} + 2] \\
\\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{23}, 0, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{sem}(s_{\mathrm{bar}}, d_{\mathrm{bar}}, t_{\mathrm{bar}}), s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{18}, 0, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{sem}(s_{\mathrm{bar}} - 1, d_{\mathrm{bar}}, t_{\mathrm{bar}}), s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) & [s_{\mathrm{bar}} \neq 0] \\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{23}, 0, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{sem}(s_{\mathrm{bar}}, d_{\mathrm{bar}}, t_{\mathrm{bar}}), s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{23}, t_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{sem}(s_{\mathrm{bar}}, d_{\mathrm{bar}}, t_{\mathrm{bar}} + 2), s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) & [s_{\mathrm{bar}} = 0] \\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{23}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{sem}(s_{\mathrm{bar}}, d_{\mathrm{bar}}, t_{\mathrm{bar}}), s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 1) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{18}, 0, n_{\mathrm{cus}}, n_{\mathrm{seat}}), \mathsf{sem}(s_{\mathrm{bar}}, d_{\mathrm{bar}}, t_{\mathrm{bar}}), s_{\mathrm{cus}}, s_{\mathrm{seat}}, \mathit{frst}, 0) & [n_{\mathrm{bar}} = d_{\mathrm{bar}} \wedge n_{\mathrm{bar}} \neq 0] \\
\\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{25}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{18}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}} + 2, t_{\mathrm{seat}}), \mathit{frst}, 1) & [t_{\mathrm{seat}} \neq d_{\mathrm{seat}} + 2] \\
\mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{25}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}}, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) & \\
\quad \to \mathsf{cnfg}(p_{\mathrm{bar}}, \mathsf{p}(\mathsf{cstmr}_{18}, n_{\mathrm{bar}}, n_{\mathrm{cus}}, 0), s_{\mathrm{bar}}, s_{\mathrm{cus}}, \mathsf{sem}(s_{\mathrm{seat}} + 1, d_{\mathrm{seat}}, t_{\mathrm{seat}}), \mathit{frst}, 0) & [t_{\mathrm{seat}} = d_{\mathrm{seat}} + 2]
\end{array}
\right\}
$$

Figure 5: LCTRS $\mathcal{R}_{\mathsf{cstmr}}$ representing the first process executing `customer` in Program 2