Bisimilarity in Fresh-Register Automata

Andrzej S. Murawski
University of Warwick

Steven J. Ramsay
University of Warwick

Nikos Tzevelekos
Queen Mary University of London

***Abstract*—Register automata are a basic model of computation over infinite alphabets. Fresh-register automata extend register automata with the capability to generate fresh symbols in order to model computational scenarios involving name creation. This paper investigates the complexity of the bisimilarity problem for classes of register and fresh-register automata. We examine all main disciplines that have appeared in the literature: general register assignments; assignments where duplicate register values are disallowed; and assignments without duplicates in which registers cannot be empty. In the general case, we show that the problem is EXPTIME-complete.**

However, the absence of duplicate values in registers enables us to identify inherent symmetries inside the associated bisimulation relations, which can be used to establish a polynomial bound on the depth of Attacker-winning strategies. Furthermore, they enable a highly succinct representation of the corresponding bisimulations. By exploiting results from group theory and computational group theory, we can then show solvability in PSPACE and NP respectively for the latter two register disciplines. In each case, we find that freshness does not affect the complexity class of the problem.

The results allow us to close a complexity gap for language equivalence of deterministic register automata. We show that deterministic language inequivalence for the no-duplicates fragment is NP-complete, which disproves an old conjecture of Sakamoto.

Finally, we discover that, unlike in the finite-alphabet case, the addition of pushdown store makes bisimilarity undecidable, even in the case of visibly pushdown storage.

I. Introduction

Register automata are one of the simplest models of computation over infinite alphabets. They consist of finite-state control and finitely many registers for storing elements from the infinite alphabet. Since their introduction by Kaminski and Francez [12] as a candidate formalism for capturing regularity in the infinite-alphabet setting, they have been actively researched especially in the database and verification communities: selected applications include the study of markup languages [17] and run-time verification [9]. While register automata can detect symbols that are currently not stored in registers (local freshness), the bounded number of registers means that they are not in general capable of recognising inputs that are genuinely fresh in the sense that they occur in the computation for the first time (global freshness). Because such a feature is desirable in many contexts, notably dynamic resource allocation, the formalism has been extended in [25] to fresh-register automata, which do account for global freshness. This paper is concerned with the problem of *bisimilarity testing* for register and fresh-register automata.

Bisimulation is a fundamental notion of equivalence in computer science. Its central role is, in part, derived from the fact that it is intensional and yet very robust. Consequently, the algorithmics of bisimilarity have attracted a lot of attention from researchers interested in the theory and practice of equivalence checking. When the set of observable actions available to a system is finite, a lot is already known about the complexity of the problem for specific classes of systems, although tight bounds are often difficult to obtain in the infinite-state cases [24]. In this paper we prove a number of bounds on the complexity of bisimulation equivalence checking. We note that in this setting language equivalence is known to be undecidable [17].

Our results are expressed using a unified framework that comprises all variations that have appeared in the literature. They differ in the allowed register assignment discipline, which turns out to affect complexity. Assignments are allowed to be: (S) *single*, if the contents of all registers are required to be distinct; or (M) *multiple*, if we allow for duplicate values. Furthermore, registers are required to: (F) always be filled; or ($\#_0$) initially allowed to be empty; or ($\#$) allowed to be erased and filled during a run[1]. The complexity of bisimilarity checking for each combination is summarised in the table below, where we use the suffix "-c" to denote completeness for this class and "-s" to denote solvability only. The results hold regardless of whether one considers register or fresh-register automata.

$(M\#)$	$(M\#_0)$	(MF)	$(S\#)$	$(S\#_0)$	(SF)
EXP-c	EXP-c	EXP-c	EXP-c	PSPACE-c	NP-s

Our work thus provides a practical motivation for modelling systems with single assignment whenever possible — if the system does not need to erase the contents of registers mid-run, the corresponding equivalence problems are lower in the complexity hierarchy.

We start by giving coarse, exponential-time upper bounds for all the classes of system considered by showing how any such bisimilarity problem can be reduced to one for finite-state automata at exponential cost. For all the multiple assignment machines this bound is tight and, for single assignment, tightness depends upon whether or not erasing is allowed. The implied significance of being able to erase the contents of registers is explained by our proof that the bisimulation games associated with such systems can simulate the computations of alternating Turing machines running in PSPACE. Here we set up an encoding of the tape, determined by the presence or absence of content in certain registers, and erasing of registers

[1]Empty content is "#". A full definition of each of the automaton variants is given in Section II.

corresponds to writing of tape cells.

Once erasure is forbidden under single assignments, we obtain better bounds by investigating the structure of the associated bisimulation relations. Such relations are generally infinite, but only the relationship between the register assignments in two configurations is relevant to bisimilarity, and so we work with a finite, though exponentially large, class of symbolic relations built over partial permutations (to link register indices). Due to the inherent symmetry and transitivity of bisimilarity, each such relation forms an inverse semigroup under function composition. Also, crucially, the relations are upward closed in the information order. Although, taken separately, neither of the preceding facts leads to an exponential leap in succinctness of representation, taken together they reveal an interconnected system of (total) permutation groups underlying each relation. What is more, in any play of the associated bisimulation game, the number of registers that are empty must monotonically decrease. This, together with an application of Babai's result on the length of subgroup chains in symmetric groups [4], allows us to show that any violation of bisimilarity can be detected after polynomially many rounds of the bisimulation game. Consequently, in this case, we are able to decide bisimilarity in polynomial space.

The polynomial bound mentioned above enables us to close a complexity gap (between NP and PSPACE) in the study of deterministic language equivalence. Namely, we show that the language inequivalence problem for *deterministic* RA($S\#_0$) is solvable in NP, and thus NP-complete, refuting a conjecture by Sakamoto [19].

Further, if registers are additionally required to be filled (SF), we can exhibit very compact representations of the relevant bisimulation relations. The fact that permutation groups have small generating sets [14] allows us then to design a representation for symbolic bisimulations that is at most polynomial in size. Furthermore, by exploiting polynomial-time membership testing for permutation groups given in terms of their generators [8], we show that such a representation can be guessed and verified by a nondeterministic Turing machine in polynomial time.

Finally, we consider bisimilarity for visibly pushdown register automata (VPDRA) under the SF register discipline, and we show that the problem here is already undecidable. Since VPDRA(SF) are a particularly weak variant, this result implies undecidability for all PDRA considered in [16]. In contrast, for finite alphabets, bisimilarity of pushdown automata is known to be decidable [22] but non-elementary [5] and, in the visibly pushdown case, EXPTIME-complete [23].

Related Work. The complexity of bisimilarity problems has been studied extensively in the finite-alphabet setting and the current state of the art for infinite-state systems is summarised nicely in [24]. Recent papers concerning the complexity of decision problems for register automata have, until now, not considered bisimulation equivalence. However, there are several related complexity results in the concurrency literature. In his PhD thesis, Pistore [18], gives an exponential-time algorithm for bisimilarity of HD-automata [15]. Since Pistore shows that bisimulation relations for HD-automata have many of the algebraic properties as the relations we study here, it seems likely that our algorithm could be adapted to show NP-solvability of the bisimilarity problem for HD-automata. Jonsson and Parrow [11] and Boreale and Trevisan [6] consider bisimilarity over a class of data-independent processes. These processes are terms built over an infinite alphabet, but the behaviour of such a process does not depend upon the data from which it is built. In the latter work, the authors also consider a class of value-passing processes, whose behaviour may depend upon the result of comparing data for equality. They show that if such processes can be defined recursively then the problem is EXPTIME-complete. Since value passing can be seen as a purely functional proxy for multiple register assignments, this result neatly reflects our findings for RA($M\#$). Finally, decidability of bisimilarity for FRA($S\#_0$) was proven in [25], albeit without a proper study of its complexity (the procedure given in *loc. cit.* can be shown to run in NEXPTIME).

Structure. In Section II we introduce the preliminaries and prove all of the EXPTIME bounds in Section III. Then we start the presentation of other results with register automata, as the addition of global freshness requires non-trivial modifications. In Section IV we show bounds for the ($S\#_0$) problems and apply the techniques to deterministic language equivalence in Section V. Section VI covers further improvements for the (SF) case. In Section VII we generalise our techniques to fresh-register automata and, finally, consider the pushdown case in Section VIII. For reasons of space, full proofs are relegated to the appendices.

II. Preliminaries

We introduce some basic notation. Given a relation $R \subseteq X \times Y$, we define $\mathsf{dom}(R) = \{x \in X \mid \exists y.(x,y) \in R\}$ and $\mathsf{rng}(R) = \{y \in Y \mid \exists x.(x,y) \in R\}$. For natural numbers $i \leq j$, we write $[i,j]$ for the set $\{i, i+1, \ldots, j\}$.

A. Bisimilarity

We define bisimulations generally with respect to a labelled transition system. As we shall see, the particular systems that we will be concerned with in this paper are the configuration graphs of various classes of (fresh-) register automata.

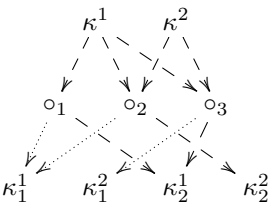
Definition 1. A ***labelled transition system*** (LTS) is a tuple $\mathcal{S} = (\mathbb{C}, \mathcal{A}ct, \{\xrightarrow{\ell} \mid \ell \in \mathcal{A}ct\})$, where $\mathbb{C}$ is a set of *configurations*, $\mathcal{A}ct$ is a set of *action labels*, and $\xrightarrow{\ell} \subseteq \mathbb{C} \times \mathbb{C}$ is a *transition relation* for each $\ell \in \mathcal{A}ct$.

A binary relation $R \subseteq \mathbb{C} \times \mathbb{C}$ is a ***bisimulation*** if for each $(\kappa_1, \kappa_2) \in R$ and each $\ell \in \mathcal{A}ct$, we have: (1) if $\kappa_1 \xrightarrow{\ell} \kappa_1'$, then there is some $\kappa_2 \xrightarrow{\ell} \kappa_2'$ with $(\kappa_1', \kappa_2') \in R$; (2) if $\kappa_2 \xrightarrow{\ell} \kappa_2'$, then there is some $\kappa_1 \xrightarrow{\ell} \kappa_1'$ with $(\kappa_1', \kappa_2') \in R$. We say that κ_1 and κ_2 are ***bisimilar***, written $\kappa_1 \sim \kappa_2$, just if there is some bisimulation R with $(\kappa_1, \kappa_2) \in R$.

Let us recall that bisimilarity has a very natural game-theoretic account. Given two configurations, one can consider

a *bisimulation game* involving two players, traditionally called *Attacker* and *Defender* respectively. They play rounds in which Attacker fires a transition from one of the configurations and Defender has to follow with an identically labelled transition from the other configuration. In the first round, the chosen transitions must lead from the configurations to be tested for bisimilarity, while, in each subsequent round, they must start at the configurations reached after the preceding round. Defender loses if he cannot find a matching transition. In this framework, bisimilarity corresponds to the existence of a winning strategy for Defender.

The process of playing a bisimulation game naturally favours Attacker as the decision maker but, thanks to the forcing technique of [10], it is possible to construct transition systems in which Defender effectively ends up making choices. By mimicking the pattern of transitions shown below, we can arrange for the Defender to force the Attacker to visit particular states during a bisimulation game. The nodes in the figure represent configurations: if the bisimulation game reaches (κ^1, κ^2), Defender can force the game to proceed to (κ^1_1, κ^2_1) or (κ^1_2, κ^2_2).



B. Fresh-register automata

We will be interested in testing bisimilarity of configurations generated by machines with registers and pushdown stack in the infinite-alphabet setting, i.e. as $\mathcal{A}ct$ we shall use the set $\Sigma \times \mathcal{D}$ for a finite alphabet Σ and an infinite alphabet $\mathcal{D}$ (with its elements sometimes called *names*), *cf.* data words [17].

Definition 2. Given a natural number r, a class of r-register assigments A is a set of functions from $[1, r]$ to $\mathcal{D} \uplus \{\#\}$. Fix such a class. An r-***fresh-register automaton*** (r-FRA) is a tuple $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F \rangle$, where:

- Q is a finite set of states, $q_0 \in Q$ initial and $F \subseteq Q$ final;
- $\rho_0 \in A$ is the initial r-register assignment;
- $\delta \subseteq Q \times \Sigma \times (\mathcal{P}([1, r]) \cup \{\circledast\}) \times [0, r] \times \mathcal{P}([1, r]) \times Q$ is the transition relation, with elements written as $q \xrightarrow{t,X,i,Z} q'$. We assume that in any such transition $i \notin Z$.

Finally an r-***register automaton*** (r-RA) is a special case of an r-FRA such that all its transitions $q \xrightarrow{t,X,i,Z} q'$ satisfy $X \neq \circledast$.

A register assignment then is just a mapping of register indices to letters from the infinite alphabet $\mathcal{D}$ and the special symbol $\#$. This symbol is used to represent the fact that a register is empty, i.e. contains no letter from $\mathcal{D}$. Consequently, by slight abuse of notation, for any r-register assignment ρ we will be writing $\mathsf{rng}(\rho)$ for the set $\rho([1, r]) \cap \mathcal{D}$, and $\mathsf{dom}(\rho)$ for $\rho^{-1}(\mathsf{rng}(\rho))$. Moreover, $\rho^{-1} = \{(d, i) \mid d \in \mathcal{D} \wedge (i, d) \in \rho\}$.

The meaning of a transition $q \xrightarrow{t,X,i,Z} q'$ is described as follows. The components t and X are a precondition: for the transition to be applicable, it must be that the next letter of the input has shape (t, a) for some $a \in \mathcal{D}$ and, moreover:

- if $X \subseteq [1, r]$ then a is already stored in exactly those registers named by X;
- if $X = \circledast$ then a is (globally) *fresh*: it has so far not appeared in the computation of $\mathcal{A}$.

If the transition applies then taking it results in changes being made to the current register assignment, namely: a is written into register i (unless $i = 0$, in which case it is not written at all) and all registers named by Z have their contents erased.

Definition 3. A ***configuration*** κ of an r-FRA $\mathcal{A}$ is a triple (q, ρ, H) consisting of a state $q \in Q$, an r-register assignment $\rho \in A$ and a finite set $H \subseteq \mathcal{D}$, called the *history*, such that $\mathsf{rng}(\rho) \subseteq H$. If $q_1 \xrightarrow{t,X,i,Z} q_2$ is a transition of $\mathcal{A}$, then a configuration (q_1, ρ_1, H_1) can make a transition to a configuration (q_2, ρ_2, H_2) accepting input (t, d), written $(q_1, \rho_1, H_1) \xrightarrow{(t,d)} (q_2, \rho_2, H_2)$, just if:

- $X = \{j \mid \rho_1(j) = d\}$, or $X = \circledast$ and $d \notin H$;
- for all $j \in [1, r]$, $\rho_2(j) = d$ if $j = i$; and $\rho_2(j) = \#$ if $j \in Z$; and $\rho_2(j) = \rho_1(j)$ otherwise;
- $H_2 = H_1 \cup \{d\}$.

We will sometimes write the set of configurations of $\mathcal{A}$ by $\mathbb{C}_\mathcal{A}$ and the induced transition relation by $\rightarrow_\mathcal{A}$. We let $\mathcal{S}(\mathcal{A})$ be the LTS $\langle \mathbb{C}_\mathcal{A}, \Sigma \times \mathcal{D}, \rightarrow_\mathcal{A} \rangle$.

On the other hand, a configuration κ of an r-RA $\mathcal{A}$ is a pair (q, ρ) of a state $q \in Q$ and an r-register assignment $\rho \in A$. The LTS $\langle \mathbb{C}_\mathcal{A}, \Sigma \times \mathcal{D}, \rightarrow_\mathcal{A} \rangle$ is defined precisely as above, albeit excluding the underlined conditions.

We define the specific classes of fresh-register automata that we will study in this work by considering specialisations of Definition 3 by the register assignment discipline followed.

Duplication in assignment. We consider two register storage policies, namely single assignment (S) or multiple assignment (M). In single assignment, we restrict the class of register assignments to be injective on non-empty registers, i.e. each $\rho \in A$ has, for all $i, j \in [1, r]$, $\rho(i) = \rho(j)$ just if $i = j$ or $\rho(i) = \# = \rho(j)$. In multiple assignment there is no such restriction. To ensure that all configurations respect the register assignment discipline, in an (S) automaton every transition $q_1 \xrightarrow{t,X,i,Z} q_2$ is required to have $X = \circledast$ or $X \subseteq \{i\}$.

Emptiness of registers. We consider the automaton's ability to process empty registers. We say that either all registers must always be filled (F), that registers may be initially empty ($\#_0$) or that the contents of registers may be erased ($\#$) during a run. Under condition (F), the class of register assignments A is restricted so that $\# \notin \mathsf{rng}(\rho)$ for each $\rho \in A$. Under conditions (F) and ($\#_0$), every transition $q_1 \xrightarrow{t,X,i,Z} q_2$ has $Z = \emptyset$ and $i \neq 0$. Condition ($\#$) imposes no specific restrictions.

We describe particular classes by the acronym FRA(XY) in which $X \in \{M, S\}$ and $Y \in \{F, \#_0, \#\}$. The class FRA(XY) is the specialisation of Definition 2 to the largest class of register assignments A satisfying the constraints imposed by X and Y. E.g. FRA($S\#_0$) are those automata whose register assignments are all functions from $[1, r]$ to $\mathcal{D} \cup \{\#\}$ that are injective on non-empty registers, and every transition of such a machine is of the form $q_1 \xrightarrow{t,X,i,Z} q_2$ with

$Z = \emptyset$, $i \neq 0$ and $X \in \{\circledast, \emptyset, \{i\}\}$. In a similar manner we define the classes RA(XY).

Remark 4. The class RA(MF) follow the register assignment discipline of the register automata defined by Segoufin [21]. The class RA($M\#_0$) follow the register assignment discipline of the M-Automata defined by Kaminski and Francez [12] and the class of RA($S\#_0$) follows the assignment discipline of the finite memory automata considered in the same paper. The class RA(SF) contain automata that follow the register assignment discipline of the machines considered by Nevin, Schwentick and Vianu [17]. The condition $i \neq 0$, which stipulates that every name encountered by the automaton be stored in some register, also originates from [12], [17]. The class FRA($S\#_0$) follow the register assignment discipline of the automata defined in [25].

In this paper we are concerned with the following family of decision problems.

Definition 5. Let $X \in \{M, S\}$ and $Y \in \{F, \#_0, \#\}$.

- The problem $\sim$-FRA(XY) is: given an FRA(XY) $\mathcal{A}$ and configurations $\kappa_1 = (q_1, \rho_1, H)$ and $\kappa_2 = (q_2, \rho_2, H)$, does $\kappa_1 \sim \kappa_2$ hold in $\mathcal{S}(\mathcal{A})$?
- The problem $\sim$-RA(XY) is: given an RA(XY) $\mathcal{A}$ and configurations κ_1 and κ_2 , does $\kappa_1 \sim \kappa_2$ hold in $\mathcal{S}(\mathcal{A})$?

We shall relate the various classes of bisimilarity problems that we study by their complexity. We write $P_1 \leq P_2$ to denote that there is a polynomial-time many-one reduction from problem P_1 to problem P_2.

Lemma 6. *The considered bisimilarity problems can be related as in Figure 1.*

C. Groups and permutations

For any $S \subseteq [1, n]$, we shall write $\mathcal{S}_S$ for the group of permutations on S, and $\mathcal{IS}_S$ for the inverse semigroup of partial permutations on S. For economy, we write $\mathcal{S}_n$ for $\mathcal{S}_{[1,n]}$; and $\mathcal{IS}_n$ for $\mathcal{IS}_{[1,n]}$.

For partial permutations σ and τ, we write $\sigma;\tau$ for their relational composition: $\sigma;\tau = \{\, (i,j) \mid \exists k.\sigma(i) = k \wedge \tau(k) = j \,\}$. Moreover, for any σ and $i, j \in [1, n]$, we let $\sigma[i \mapsto j]$ be the result of updating σ with (i, j):

$$\sigma[i \mapsto j] \;=\; \{(i,j)\} \cup \{(k, \sigma(k)) \mid k \neq i \wedge \sigma(k) \neq j\}.$$

For all $j \in [1, n]$, $\sigma \in \mathcal{S}_S$ and $S \subseteq [1, n]$ we also write $S[j]$ for $S \cup \{j\}$, and $\sigma \cdot S$ for $\{\sigma(i) \mid i \in S\}$.

III. Bisimilarity problems complete for EXPTIME

In this section we show that the upper four classes in our two hierachies of automata all have bisimilarity problems that are complete for exponential time.

Theorem 7. *All of the problems $\sim$-RA($S\#$), $\sim$-RA(MF), $\sim$-RA($M\#_0$), $\sim$-RA($M\#$), $\sim$-FRA($S\#$), $\sim$-FRA(MF), $\sim$-FRA($M\#_0$) and $\sim$-FRA($M\#$) are EXPTIME-complete.*

Proof: The result follows immediately from Propositions 8 and 10 and Lemma 6. ∎

Our argument proceeds by showing that $\sim$-FRA($M\#$) can be solved in EXPTIME (Proposition 8) and $\sim$-RA($S\#$) is already EXPTIME-hard (Proposition 10). For the former, we reduce the problem to a bisimilarity problem for finite state automata of exponential size.

Given an instance of the r-register, FRA($M\#$) bisimilarity problem, the idea is to construct a bisimilarity problem for a finite automaton over an alphabet derived from a finite subset $N \subseteq \mathcal{D}$ of size $2r + 2$. Given a configuration $\kappa = (q, \rho, H)$ of the FRA, we represent it by an abstract configuration $\phi \cdot \kappa = (q, \phi \cdot \rho, \phi \cdot H)$ which is built entirely from letters in N. Here $\phi : \mathcal{D} \to N$ is surjective, $\phi \cdot \rho = (\phi[\# \mapsto \#]) \circ \rho$ and $\phi \cdot H = \{\phi(d) \mid d \in H\}$. We choose the abstraction ϕ in such a way that it partitions $\mathcal{D}$ and N with respect to $\mathsf{rng}(\rho)$ and H: that is, $\phi = \phi_1 \uplus \phi_2 \uplus \phi_3$ where $\mathsf{rng}(\phi_i)$ are all distinct and $\mathsf{dom}(\phi_1) = \mathsf{rng}(\rho)$, $\mathsf{dom}(\phi_2) = H \setminus \mathsf{rng}(\rho)$ and $\mathsf{dom}(\phi_3) = \mathcal{D} \setminus H$. In addition, ϕ_1 is injective.

The partitioning conditions ensure that our representation of configurations by abstract configurations is faithful. But, due to global freshness, the abstraction ϕ cannot be chosen uniformly for the entire simulation. This is because, with the alphabet limited to N, there would be no letters available to be played as part of globally fresh transitions as soon as the simulated history $\phi \cdot H$ became equal to N. Hence, the simulation needs to recycle letters in the history as soon as they become otherwise irrelevant to the current configuration and, consequently, a new (typically smaller) history and a new abstraction ϕ' must be chosen at each step to reflect this. However, at position $(q_1, \phi \cdot \rho_1, \phi \cdot H), (q_2, \phi \cdot \rho_2, \phi \cdot H)$ of the simulation, the only letters that can be recycled are those that are not in $\phi \cdot \rho_1$ or $\phi \cdot \rho_2$. Recycling such a letter d by removing it from $\phi \cdot H$ is unfaithful to the simulation, since it would potentially allow a globally fresh transition playing d to be matched by a local one. This demonstrates that it is necessary to know *both* register assignments of the position in order to choose which letters are available to recycle and hence the shape of a new history.

To this end, the bisimulation game induced by the simulating finite automaton is constructed so that both of the two component systems contain both of $\phi \cdot \rho_1$ and $\phi \cdot \rho_2$.

Proposition 8. *$\sim$-FRA($M\#$) is solvable in EXPTIME.*

Proof sketch: Given an r-FRA($M\#$) $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F\rangle$ and a pair of input configurations, we decide their bisimilarity by checking an equivalent bisimilarity problem on a finite-state automaton $\mathcal{B}$. Each state of the finite automaton is of the form $(\sigma, q_1, \rho_1, H, q_2, \rho_2, p)$, where ρ_1 and ρ_2 are r-register assignments drawn only from N, representing the left and right component systems of the bisimulation game that is being simulated. States q_1 and q_2 are from Q, H is a history built only over N and σ and p are bookeeping information drawn from sets of constant size. The construction induces a bisimulation game in which one turn of the original game is simulated in two parts, consisting of four turns.

In the first part, Attacker announces which component he would like to play from ($i \in \{1, 2\}$) and then which transition

$$\begin{array}{ccccccccccc}
\sim\text{-FRA}(SF) & \leq & \sim\text{-FRA}(S\#_0) & \leq & \sim\text{-FRA}(S\#) & \leq & \sim\text{-FRA}(MF) & \leq & \sim\text{-FRA}(M\#_0) & \leq & \sim\text{-FRA}(M\#) \\
\vee\!| & & \vee\!| & & \vee\!| & & \vee\!| & & \vee\!| & & \vee\!| \\
\sim\text{-RA}(SF) & \leq & \sim\text{-RA}(S\#_0) & \leq & \sim\text{-RA}(S\#) & \leq & \sim\text{-RA}(MF) & \leq & \sim\text{-RA}(M\#_0) & \leq & \sim\text{-RA}(M\#)
\end{array}$$

Fig. 1. Relationship between the main bisimilarity problems considered in this work.

$T \in \delta$ he would like to simulate and which letter from N he wishes to simulate it with. He then updates register assignment ρ_i in the state accordingly, leaving ρ_{3-i} untouched. Defender responds by announcing the same transition and letter and updating the same register assignment ρ_i but in the other component.

In the second part, Defender uses Defender forcing in order to choose which transition she would like to simulate in response to Attacker's choice T, the letter used to simulate it (which must be the same as the one chosen by Attacker) and what the value of the new history should be. She then updates register assignment ρ_{3-i} and the history in her component accordingly. Attacker is forced to respond by announcing the same transition, letter and choice of history and updating assignment ρ_{3-i} and the history in his component to match.

It can be shown that such a simulation gives a faithful reduction from the FRA bisimilarity problem to the bisimilarity problem for finite automata. The number of states of the automaton this way constructed is bounded by $O(|Q| \cdot 2^{|N|+2r\log|N|})$. The alphabet of the finite automaton, whose letters simultaneously announce a transition of the FRA, a letter from N and a history, are bounded above by $O(|\delta| \cdot 2^{|N|})$. Hence bisimilarity can be solved in time $O(|\delta| \cdot |Q|^2 \cdot 2^{3|N|+4r|N|\log|N|})$. Since $|N|$ is $O(r)$, it follows that bisimilarity can be decided in time exponential in the size of the FRA, or polynomial in the size of the FRA for a fixed value of r. ∎

Remark 9. The preceding proof shows that one turn of the FRA($M\#$) bisimulation game can be simulated by using four turns of the bisimulation game for the simulating finite automaton. Consequently, any winning strategy for the FRA-induced game can be transformed into a winning strategy for the finite automaton-induced game with at most a constant-factor increase in depth.

Further down the hierachy, to show $\sim$-RA($S\#$) is EXPTIME-hard, we use the registers of this class of automata to represent the tape content of bounded Turing machines.

Proposition 10. *$\sim$-RA($S\#$) is EXPTIME-hard.*

Proof sketch: We reduce instances of the Alternating Linear Bounded Automaton (ALBA) acceptance problem, which is known to be EXPTIME-hard, to RA($S\#$) bisimilarity. From an ALBA $\mathcal{M}$ we construct an RA($S\#$) $\mathcal{A}$ that simulates it, with the the binary tape content of $\mathcal{M}$ encoded by the register assignment of $\mathcal{A}$. At every step of the bisimulation game, we arrange for Defender to choose transitions from existential states (using Defender forcing [10]) and Attacker to choose from universal states.

The ability of $\mathcal{A}$ to use empty registers and to erase full registers is key to encoding the exponential amount of information that is potentially held on the tape in a number of registers that is polynomial in its size. To this end, to represent a tape of size n, we equip $\mathcal{A}$ with $2n$ registers, under the following encoding. Cell k of the tape has 0 written on it iff register $2k-1$ is empty and it has 1 written on it iff register $2k$ is empty. ∎

IV. PSPACE-completeness for RAs with single assignment without erasure (RA($S\#_0$))

We next prove that the EXPTIME bound can be improved if duplicate values and erasures are forbidden. We handle register automata first to expose the flavour of our technique. The main result is given below. It will follow from Propositions 21 and 22.

Theorem 11. *$\sim$-RA($S\#_0$) is PSPACE-complete.*

Simplified notation: Recall that, in any transition $q_1 \xrightarrow{t,X,i,Z} q_2$ of an r-RA($S\#_0$), we have that $Z = \emptyset$, $i \neq 0$ and $X \subseteq \{i\}$. These restrictions allow for a simpler notation for transitions, with $\delta \subseteq Q \times \Sigma \times ([1,r] \cup \{ i^\bullet \mid i \in [1,r] \}) \times Q$:

(a) we write each transition $q_1 \xrightarrow{t,\{i\},i,\emptyset} q_2$ as $q_1 \xrightarrow{t,i} q_2$;
(b) and each transition $q_1 \xrightarrow{t,\emptyset,i,\emptyset} q_2$ as $q_1 \xrightarrow{t,i^\bullet} q_2$.

Thus, transitions of type (a) correspond to the automaton reading an input (t,a) where a is the name in the i-th register; while in (b) transitions the automaton reads (t,a) if a is *locally fresh*, that is, it does not appear in the registers, and in this case a will be stored in register i.

A. Symbolic bisimulation

We attack the bisimulation problem *symbolically*, i.e. by abstracting actual names in the bisimulation game to the indices of the registers where these names reside. This will lead us to consider groups of finite permutations and inverse semigroups of partial finite permutations[2]. We shall define symbolic bisimulations over pairs (q,S) of a state q and a set of register indices $S \subseteq [1,r]$. In this way, the locations of the empty registers $[1,r] \setminus S$ are made explicit.

Definition 12. Let $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F \rangle$ be an r-RA($S\#_0$). We first set:

$$\begin{aligned}
\mathcal{U}_0 &= Q \times \mathcal{P}([1,r]) \times \mathcal{IS}_r \times Q \times \mathcal{P}([1,r]) \\
\mathcal{U} &= \{ (q_1, S_1, \sigma, q_2, S_2) \in \mathcal{U}_0 \mid \sigma \subseteq S_1 \times S_2 \}
\end{aligned}$$

[2]Recall that an inverse semigroup generalises a group in the sense that an inverse semigroup may not have a unit element, and multiplying an element with its inverse does not yield a unit.

A *symbolic simulation* on $\mathcal{A}$ is a relation $R \subseteq \mathcal{U}$, with elements $(q_1, S_1, \sigma, q_2, S_2) \in R$ written infix $(q_1, S_1)\, R_\sigma\, (q_2, S_2)$, such that all $(q_1, S_1, \sigma, q_2, S_2)$ satisfy the following *symbolic simulation conditions* (SYS)[3]:

- for all $q_1 \xrightarrow{t,i} q_1'$,
  - if $i \in \mathsf{dom}(\sigma)$ then there is some $q_2 \xrightarrow{t,\sigma(i)} q_2'$ with $(q_1', S_1)\, R_\sigma\, (q_2', S_2)$,
  - if $i \in S_1 \setminus \mathsf{dom}(\sigma)$ then there is some $q_2 \xrightarrow{t,j^\bullet} q_2'$ with $(q_1', S_1)\, R_{\sigma[i\mapsto j]}\, (q_2', S_2[j])$;
- for all $q_1 \xrightarrow{t,i^\bullet} q_1'$,
  - there is some $q_2 \xrightarrow{t,j^\bullet} q_2'$ with $(q_1', S_1[i])\, R_{\sigma[i\mapsto j]}\, (q_2', S_2[j])$,
  - for all $j \in S_2 \setminus \mathsf{rng}(\sigma)$, there is some $q_2 \xrightarrow{t,j} q_2'$ with $(q_1', S_1[i])\, R_{\sigma[i\mapsto j]}\, (q_2', S_2)$.

We let the inverse of R be

$$R^{-1} = \{\, (q_2, S_2, \sigma^{-1}, q_1, S_1) \mid (q_1, S_1, \sigma, q_2, S_2) \in R \,\}$$

and call R a ***symbolic bisimulation*** if both R and R^{-1} are symbolic simulations. We let *s-bisimilarity*, denoted $\overset{\mathsf{s}}{\sim}$, be the union of all symbolic bisimulations. We say that configurations (q_1, ρ_1) and (q_2, ρ_2) are *s-bisimilar*, written $(q_1, \rho_1) \overset{\mathsf{s}}{\sim} (q_2, \rho_2)$, if $(q_1, \mathsf{dom}(\rho_1)) \overset{\mathsf{s}}{\sim}_{\rho_1;\rho_2^{-1}} (q_2, \mathsf{dom}(\rho_2))$.

We approximate symbolic bisimilarity by a sequence of ***indexed bisimilarity*** relations $\overset{i}{\sim} \subseteq \mathcal{U}$ defined inductively as follows. First, we let $\overset{0}{\sim}$ be the whole of $\mathcal{U}$. Then, for all $i \in \omega$, $(q_1, S_1) \overset{i+1}{\sim}_\tau (q_2, S_2)$ just if $(q_1, S_1, \tau, q_2, S_2)$ and $(q_2, S_2, \tau^{-1}, q_1, S_1)$ both satisfy the (SYS) conditions in $\overset{i}{\sim}$.

Lemma 13. *Let (q_1, ρ_1), (q_2, ρ_2) be configurations of an r-RA($S\#_0$), then: $(q_1, \rho_1) \sim (q_2, \rho_2) \iff (q_1, \rho_1) \overset{\mathsf{s}}{\sim} (q_2, \rho_2)$. Furthermore, for all $i \in \omega$, $\overset{i+1}{\sim} \subseteq \overset{i}{\sim}$ and $(\bigcap_{i\in\omega} \overset{i}{\sim}) = \overset{\mathsf{s}}{\sim}$.*

Our next aim is to show that $\overset{\mathsf{s}}{\sim}$ and each $\overset{i}{\sim}$ are closed under composition and extension of partial permutations. The latter allows us, in Lemma 18, to bound the convergence of the indexed bisimulations by finding within them strict chains of subgroups. The former, in Section VI, helps us to represent $\overset{\mathsf{s}}{\sim}$ succinctly by appropriate choices of representatives.

Given $S_1, S_2 \subseteq [1, r]$ and $\sigma, \sigma' \in \mathcal{IS}_r$ we write $\sigma \leq_{S_1,S_2} \sigma'$ just if $\sigma \subseteq \sigma' \subseteq S_1 \times S_2$. Moroever, given $X \subseteq S \subseteq [1, r]$, we write id_X for the partial map from S to S that acts as identity on X (and is undefined otherwise). For any $R \subseteq \mathcal{U}$, we define its ***closure*** $Cl(R)$ to be the smallest relation R' containing R and closed under the following rules.

$$\frac{}{(q, S, \mathsf{id}_S, q, S) \in R'}\ (\textsc{Id}) \qquad \frac{(q_1, S_1, \sigma, q_2, S_2) \in R'}{(q_2, S_2, \sigma^{-1}, q_1, S_1) \in R'}\ (\textsc{Sym})$$

$$\frac{(q_1, S_1, \sigma, q_2, S_2) \in R' \qquad \sigma \leq_{S_1,S_2} \sigma'}{(q_1, S_1, \sigma', q_2, S_2) \in R'}\ (\textsc{Ext})$$

$$\frac{(q_1, S_1, \sigma_1, q_2, S_2) \in R' \qquad (q_2, S_2, \sigma_2, q_3, S_3) \in R'}{(q_1, S_1, \sigma_1; \sigma_2, q_3, S_3) \in R'}\ (\textsc{Tr})$$

[3]We say that $(q_1, S_1, \sigma, q_2, S_2)$ *satisfies the* (SYS) *conditions in* R.

We say R is *closed* in case $Cl(R) = R$. We can show:

Lemma 14. *Let $P, R \subseteq \mathcal{U}$ be such that $R = R^{-1}$. If all $g \in R$ satisfy the* (SYS) *conditions in P then all $g \in Cl(R)$ satisfy the* (SYS) *conditions in $Cl(P)$.*

Much of the following development relies upon the fact that bisimilarity and indexed bisimilarity have a closed structure.

Corollary 15. *(Closures) Bisimilarity and indexed bisimilarity for RA($S\#_0$) are both closed:*
1) $\overset{\mathsf{s}}{\sim} = Cl(\overset{\mathsf{s}}{\sim})$; 2) for all $i \in \omega$: $\overset{i}{\sim} = Cl(\overset{i}{\sim})$.

Proof: For 1 note that $\overset{\mathsf{s}}{\sim} = (\overset{\mathsf{s}}{\sim})^{-1}$ and all its elements satisfy the (SYS) conditions in $\overset{\mathsf{s}}{\sim}$. Hence, by Lemma 14 we have that $Cl(\overset{\mathsf{s}}{\sim})$ is a symbolic bisimulation, i.e. $Cl(\overset{\mathsf{s}}{\sim}) = \overset{\mathsf{s}}{\sim}$. The result then follows. For 2 we proceed by induction on i. When $i = 0$ then the result follows from the fact that $\overset{0}{\sim}$ is the universal relation. For the inductive case, note first that $\overset{i+1}{\sim}$ is symmetric by construction and all $g \in \overset{i+1}{\sim}$ satisfy the (SYS) conditions in $\overset{i}{\sim}$. Hence, by Lemma 14, all elements of $Cl(\overset{i+1}{\sim})$ satisfy the (SYS) conditions in $Cl(\overset{i}{\sim})$. By IH, $Cl(\overset{i}{\sim}) = \overset{i}{\sim}$ so $Cl(\overset{i+1}{\sim}) \subseteq \overset{i+1}{\sim}$, as required. ∎

B. Permutation groups

Next we present a series of results that uncover group-theoretic structure in closed relations. Given $p \in Q$, $S \subseteq [1, r]$ and R closed, let $\mathcal{J}_S^p(R) = \{X \mid X \subseteq S,\ (p, S)\ R_{\mathsf{id}_X}\ (p, S)\}$.

Lemma 16. *$\mathcal{J}_S^p(R) \neq \emptyset$ and if $X_1, X_2 \in \mathcal{J}_S^p(R)$ then $X_1 \cap X_2 \in \mathcal{J}_S^p(R)$.*

Proof: $\mathcal{J}_S^p(R) \neq \emptyset$ follows from $S \in \mathcal{J}_S^p(R)$. For the rest, we observe that $\mathsf{id}_{X_1}; \mathsf{id}_{X_2} = \mathsf{id}_{X_1 \cap X_2}$ and R is closed. ∎

It follows from the lemma above that $\mathcal{J}_S^p(R)$ contains the least element with respect to inclusion, which we shall call *the* ***characteristic set*** *of (p, S) in R* and denote by $X_S^p(R)$. By Corollary 15, $\mathcal{J}_S^p(R) = \{X \mid X_S^p(R) \subseteq X \subseteq S\}$.

The family $\{X_S^p(R)\}_{p\in Q}$ turns out to play an important structural role in R for the following reason.

Lemma 17. *Let $p \in Q$ and $\mathcal{G}_S^p(R) = \{\sigma \cap (X_S^p(R) \times X_S^p(R)) \mid (p, S)\ R_\sigma\ (p, S)\}$. Then $\mathcal{G}_S^p(R)$ is a group (under composition). In particular, it is a subgroup of $\mathcal{S}_{X_S^p(R)}$.*

Proof (sketch): First, since $(p, S)\ R_{\mathsf{id}_S}\ (p, S)$, we have $\mathsf{id}_{X_S^p(R)} \in \mathcal{G}_S^p(R)$. Now, $(p, S)\ R_\sigma\ (p, S)$ implies $(p, S)\ R_{\sigma^{-1}}\ (p, S)$. The existence of inverses is proved by establishing that $\sigma \cap (X_S^p(R) \times X_S^p(R))$ and $\sigma^{-1} \cap (X_S^p(R) \times X_S^p(R))$ are bijective. Thus, $(\sigma \cap (X_S^p(R) \times X_S^p(R)))\,;(\sigma^{-1} \cap (X_S^p(R) \times X_S^p(R))) = \mathsf{id}_{X_S^p(R)}$. ∎

Since indexed bisimulations are closed, they have group-theoretic structure. We use it to help estimate their rate of convergence. Recall $\mathcal{U} = Q \times \mathcal{P}([1, r]) \times \mathcal{IS}_r \times Q \times \mathcal{P}([1, r])$.

Lemma 18. *Let $S_1, S_2 \subseteq [1, r]$ and $\mathcal{U}_{S_1,S_2} = Q \times \{S_1, S_2\} \times \mathcal{IS}_r \times Q \times \{S_1, S_2\}$. Then the sub-chain $\{\overset{i}{\sim} \mid (\overset{i+1}{\sim} \cap\, \mathcal{U}_{S_1,S_2}) \subsetneq (\overset{i}{\sim} \cap\, \mathcal{U}_{S_1,S_2})\}$ has size $O(|Q|^2 + r^2|Q|)$.*

Proof (sketch): We show that changes in $\overset{j}{\sim} \cap \mathcal{U}_{S_1,S_2}$ (as j increases) can be traced back to either shrinkage of a characteristic set $X^p_S(\overset{j}{\sim})$ ($S \in \{S_1, S_2\}$), or shrinkage of $\mathcal{G}^p_S(\overset{j}{\sim})$ ($S \in \{S_1, S_2\}$) or disappearance of all tuples $(q_1, S'_1, \sigma, q_2, S'_2)$ for some $q_1, q_2 \in Q$ and $S'_1, S'_2 \in \{S_1, S_2\}$. The number of changes of each kind can be bounded by a polynomial. In the second case we appeal to the fact that strict chains of subgroups of a symmetric group on n-elements have length at most linear in n, which is a result of Babai [4]. ∎

Note that it does not quite follow from the above result that the sequence $(\overset{i}{\sim})$ converges in polynomially many steps, because there are exponentially many pairs (S_1, S_2). Next we shall establish such a bound by studying more closely the overlap in evolutions of different (S_1, S_2). Let us write $\gamma(S_1, S_2)$ for $|S_1| + |S_2|$, i.e. $0 \leq \gamma(S_1, S_2) \leq 2r$.

Lemma 19. *Let $\mathcal{U}^-_{S_1,S_2} = Q \times \{S_1\} \times \mathcal{IS}_r \times Q \times \{S_2\}$ and let c be the constant of $O(|Q|^2 + r^2|Q|)$ in Lemma 18 (2).*

1) *Then, for any (S_1, S_2), we have $\overset{j}{\sim} \cap \mathcal{U}^-_{S_1,S_2} = \overset{\mathsf{s}}{\sim} \cap \mathcal{U}^-_{S_1,S_2}$, where $j = c(2r - \gamma(S_1, S_2) + 1)(|Q|^2 + r^2|Q|)$.*
2) *Let $B = c(2r + 1)(|Q|^2 + r^2|Q|)$. For any (S_1, S_2), $\overset{B}{\sim} \cap \mathcal{U}^-_{S_1,S_2} = \overset{\mathsf{s}}{\sim} \cap \mathcal{U}^-_{S_1,S_2}$.*

Proof: For Part 1 we reason by induction on $(2r - \gamma(S_1, S_2))$. We tackle the inductive step first. Assume the result holds for all (S'_1, S'_2) with $\gamma(S'_1, S'_2) > \gamma(S_1, S_2)$. Let $j' = c(2r - (\gamma(S_1, S_2) + 1) + 1)(|Q|^2 + r^2|Q|) = c(2r - \gamma(S_1, S_2))(|Q|^2 + r^2|Q|)$. Then, for all such (S'_1, S'_2), $(\overset{j'}{\sim} \cap \mathcal{U}^-_{S'_1,S'_2}) = (\overset{\mathsf{s}}{\sim} \cap \mathcal{U}^-_{S'_1,S'_2})$.

Observe that, for $k > j'$, if $\overset{k}{\sim} \cap \mathcal{U}^-_{S_1,S_2} = \overset{k+1}{\sim} \cap \mathcal{U}^-_{S_1,S_2}$, then we must have $\overset{k}{\sim} \cap \mathcal{U}^-_{S_1,S_2} = \overset{\mathsf{s}}{\sim} \cap \mathcal{U}^-_{S_1,S_2}$, because the (SYS) conditions for (S_1, S_2) refer to either (S_1, S_2) or (S'_1, S'_2) with $\gamma(S'_1, S'_2) > \gamma(S_1, S_2)$.Consequently, if $\overset{j'}{\sim} \cap \mathcal{U}^-_{S_1,S_2} \neq \overset{\mathsf{s}}{\sim} \cap \mathcal{U}^-_{S_1,S_2}$, the sequence $(\overset{k}{\sim} \cap \mathcal{U}^-_{S_1,S_2})$ $(k = j', j' + 1, \cdots)$ will have to change in every step before stabilisation. Thus, the steps before stabilisation will induce a subchain of the chain analysed in Lemma 18 (2). Hence, at most $c(|Q|^2 + r^2|Q|)$ extra steps from $(\overset{j'}{\sim})$ will be required to arrive at $\overset{\mathsf{s}}{\sim} \cap \mathcal{U}^-_{S_1,S_2}$, which delivers the required bound.

The base case ($\gamma(S_1, S_2) = 2r$) can be established in a similar fashion: in this case the (SYS) conditions can only refer to (S_1, S_2), thus the sequence $(\overset{k}{\sim} \cap \mathcal{U}^-_{S_1,S_2})$ $(k \geq 0)$ will be strictly decreasing before stabilisation and the bound from Lemma 18 (2) can be applied.

Part 2 follows from Part 1, because $c(2r+1)(|Q|^2+r^2|Q|)$ is the largest of all the bounds. ∎

Proposition 20. *For any RA($S\#_0$) bisimulation problem, if there is a winning strategy for Attacker then there is one of depth $O(r|Q|^2 + r^3|Q|)$.*

Proof: We first observe that bisimulation strategies and their corresponding symbolic bisimulation strategies have the same depth. Thus, it suffices to bound symbolic strategies for Attacker. The $O(r|Q|^2 + r^3|Q|)$ bound follows from Part 2 of the preceding Lemma. ∎

Proposition 21. *$\sim$-RA($S\#_0$) is solvable in PSPACE.*

Proof: In Remark 9 we established that bisimilarity for RA($M\#$) can be reduced to the finite-alphabet case at the cost of prolonging the bisimulation game by a constant factor. Consequently, the polynomial bound from the preceding Proposition (for RA($S\#_0$)) is also valid after the reduction to the finite-alphabet case.

Thanks to the bound, it suffices to play the corresponding bisimulation games for polynomially many steps. The existence of a winning strategy can then be established by an alternating Turing machine running in polynomial time. The PSPACE bounds then follows from APTIME = PSPACE. ∎

Proposition 22. *$\sim$-RA($S\#_0$) is PSPACE-hard.*

Proof (sketch): We reduce from the well-known PSPACE-complete problem of checking validity of totally quantified boolean formulas in prenex conjunctive normal form. Universal quantification and selection of conjuncts is performed by Attacker. For existential quantification and disjunctions, we rely on Defender Forcing. The choices of truth values by both players are recorded in registers by using, for each variable x_i, registers $2i, 2i + 1$, both initialised to $\#$. If a player chooses *true* for x_i, we fill register $2i$ leaving $2i + 1$ empty; we do the opposite otherwise. This makes it possible to arrange for bisimilarity/non-bisimilarity (as appropriate) in the final stage of the game, depending on whether the resulting literal is negated. ∎

V. LANGUAGE EQUIVALENCE FOR $RA(S\#_0)$

The results of the previous section can be used to close an existing complexity gap for deterministic language equivalence of register automata. Recall that, in the non-deterministic case, language equivalence (even universality) is undecidable [17]. In the deterministic case, however, the problem can be solved in PSPACE. Sakamoto [19] conjectured that the language inequivalence problem is not in NP. Below we refute the conjecture, showing that, for RA($S\#_0$), the complexity of deterministic language inequivalence actually matches that of nonemptiness [20].

We call an r-RA($S\#_0$) $\mathcal{A}$ *deterministic* if, for all states q of $\mathcal{A}$: (i) for all $(t, i) \in \Sigma \times [1, r]$ there is at most one transition of the form $q \xrightarrow{t,i} q'$, and (ii) for all $t \in \Sigma$ there is at most one transition of the form $q \xrightarrow{t,i^\bullet} q'$. On the other hand, an LTS is deterministic if, for all $\kappa \in \mathbb{C}$ and $\ell \in \mathcal{A}ct$, there is at most one transition $\kappa \xrightarrow{\ell} \kappa'$. Note that if $\mathcal{A}$ is deterministic then so is its transition system $\mathcal{S}(\mathcal{A})$.[4] Then, from Proposition 20, one obtains the following.

Lemma 23. *Let $\mathcal{A}_i = \langle Q_i, q_{0i}, \rho_{0i}, \delta_i, F_i \rangle$ be a deterministic r_i-RA($S\#_0$) ($i = 1, 2$), $r = \max(r_1, r_2)$ and $N = |Q_1| + |Q_2|$.*

[4]The converse may fail due to transitions of $\mathcal{A}$ not being fireable in $\mathcal{S}(\mathcal{A})$.

If $\mathcal{L}(\mathcal{A}_1) \neq \mathcal{L}(\mathcal{A}_2)$ then there is some $w \in (\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)) \setminus (\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2))$ with $|w| \in O(rN^2 + r^3N)$.

Theorem 24. *Language inequivalence for deterministic RA($S\#_0$) is NP-complete.*

Proof: Membership in NP is achieved via Lemma 23. NP-hardness follows from NP-completeness of language non-emptiness for deterministic RA($S\#_0$) [20]. ∎

VI. NP BOUND FOR SINGLE ASSIGNMENT WITH FILLED REGISTERS (RA(SF))

In Section IV we showed, in the setting with single assignment and no erasures (denoted by RA($S\#_0$)) the bisimilarity problem was solvable in polynomial space. Here we show that a further improvement is possible in the RA(SF) case, i.e. if the registers are required to be filled from the very start. We shall show an NP upper bound.

We start off with a series of results aiming to identify succinct (polynomial-size) sets of generators for $\overset{\mathsf{s}}{\sim}$, which we shall call *generating systems*. In Section IV we already found that parts of $\overset{\mathsf{s}}{\sim}$ exhibit group-theoretic structure. Namely, Lemma 17 shows that, for any $p \in Q$ and $S \subseteq [1,r]$, $\mathcal{G}^p_S(\overset{\mathsf{s}}{\sim}) = \{\sigma \cap (X^p_S \times X^p_S) \,|\, (p,S) \overset{\mathsf{s}}{\sim}_\sigma (p,S)\}$ is a group, where $X^p_S(\overset{\mathsf{s}}{\sim}) \subseteq S$ is the characteristic set of (p,S).

Note that, for RA(SF), we only have the case $S = [1,r]$. Furthermore, $\overset{\mathsf{s}}{\sim}$ will be the only closed relation that we shall consider. For these reasons, we write simply X^p for characteristic set $X^p_{[1,r]}(\overset{\mathsf{s}}{\sim})$ and $\mathcal{G}^p$ for group $\mathcal{G}^p_{[1,r]}(\overset{\mathsf{s}}{\sim})$.

The group-theoretic structure implies that $\mathcal{G}^p$ can be generated by linearly many generators with respect to r.

Lemma 25 ([14])**.** *Every subgroup of $\mathcal{S}_n$ has at most $\max(2, \lfloor \frac{n}{2} \rfloor)$ generators.*

To handle the more general case $(p,S) \overset{\mathsf{s}}{\sim}_\sigma (q,S)$ of different states, consider

$$\mathcal{K}^{p,q} = \{\sigma \cap (X^p \times X^q) \mid (p,[1,r]) \overset{\mathsf{s}}{\sim}_\sigma (q,[1,r])\}.$$

Observe that, for $\sigma_1, \sigma_2 \in \mathcal{K}^{p,q}$, we have $\sigma_2 = (\sigma_2; \sigma_1^{-1}); \sigma_1$, because $\sigma_1^{-1}; \sigma_1 = \mathsf{id}_{X^q}$. Moreover, $\sigma_2; \sigma_1^{-1} \in \mathcal{G}^p$. Consequently, in presence of generators of $\mathcal{G}^p$, one member of $\mathcal{K}^{p,q}$ suffices to generate the whole of $\mathcal{K}^{p,q}$ by composition. This observation motivates the following definition of a generating system.

Definition 26. A ***generating system*** $\mathcal{G}$ consists of:

- a partitioning of Q into $P_1, \cdots, P_k$;
- for each partition P_i, a single representative $p_i \in P_i$ and:
  - a characteristic set $X^{p_i} \subseteq [1,r]$;
  - a set G^{p_i}, of up to $\max(2, \lfloor \frac{r}{2} \rfloor)$ permutations $\sigma \in \mathcal{S}_{X^{p_i}}$;
  - for each $q \in P_i \setminus \{p_i\}$, a partial permutation $\mathsf{ray}^{p_i}_q \in \mathcal{IS}_{[1,r]}$ such that $\mathsf{dom}(\mathsf{ray}^{p_i}_q) = X^{p_i}$; for technical convenience, we also add $\mathsf{ray}^{p_i}_{p_i} = \mathsf{id}_{X^{p_i}}$.

We write $rep(\mathcal{G})$ for the set $\{p_1, \cdots, p_k\}$ of representatives.

A generating system is used to generate a relation $gen(\mathcal{G}) \subseteq (Q \times \{[1,r]\} \times \mathcal{IS}_r \times Q \times \{[1,r]\})$ as follows. First, set

$$\begin{aligned}\mathrm{BASE}_\mathcal{G} = &\{(p_i,[1,r],\sigma,p_i,[1,r]) \,|\, p_i \in rep(\mathcal{G}), \sigma \in G^{p_i}\} \\ &\cup \{(p_i,[1,r],\mathsf{ray}^{p_i}_q,q,[1,r]) \,|\, p_i \in rep(\mathcal{G}), q \in P_i\}\end{aligned}$$

and then take $gen(\mathcal{G}) = Cl(\mathrm{BASE}_\mathcal{G})$.

Lemma 27. *There exists a generating system $\mathcal{G}$ such that $gen(\mathcal{G}) = \overset{\mathsf{s}}{\sim}$.*

Proof: We partition Q into equivalence classes defined by: $p \sim q$ if and only if there exists σ such that $(p,[1,r],\sigma,q,[1,r]) \in \overset{\mathsf{s}}{\sim}$. For each equivalence class P_i, we pick a single member p_i arbitrarily and let G^{p_i} consist of the generators of $\mathcal{G}^{p_i}$ provided by Lemma 25. Consider $q \in P_i \setminus \{p_i\}$. Because $q \in P_i$, there exists σ such that $(p_i,[1,r],\sigma,q,[1,r]) \in \overset{\mathsf{s}}{\sim}$. Then we can take $\mathsf{ray}^{p_i}_q = \sigma \cap (X^{p_i} \times [1,r])$. By the previous discussion, this delivers the sought generating system. ∎

Lemma 28. *For any generating system $\mathcal{G}$, membership in $gen(\mathcal{G})$ can be determined in polynomial time.*

Proof: To determine whether $(q_1,[1,r],\sigma,q_2,[1,r]) \in gen(\mathcal{G})$, we proceed as follows. If q_1, q_2 belong to different partitions we return NO. Suppose $q_1, q_2 \in P_i$. Recall that $\mathrm{BASE}_\mathcal{G}$ contains $(p_i,[1,r],\mathsf{ray}^{p_i}_{q_j},q_j,[1,r])$ with $\mathsf{dom}(\mathsf{ray}^{p_i}_{q_j}) = X^{p_i}$. Then $(q_1,[1,r],\sigma,q_2,[1,r]) \in gen(\mathcal{G})$ is equivalent to $(p_i,[1,r],\sigma',p_i,[1,r]) \in gen(\mathcal{G})$, where $\sigma' = \mathsf{ray}^{p_i}_{q_1}; \sigma; (\mathsf{ray}^{p_i}_{q_2})^{-1}$. This is in turn equivalent to $\sigma' \cap (X^{p_i} \times X^{p_i})$ being generated from permutations in G^{p_i}. That the latter problem is solvable in polynomial time is a well-known result in computational group theory [8]. ∎

Theorem 29. *$\sim$-RA(SF) is solvable in NP.*

Proof: First we guess a generating system $\mathcal{G}$ and verify whether $gen(\mathcal{G})$ is a bisimulation. By Lemma 27, there exists at least one generating system with this property. Because generating systems involve polynomially many components of polynomial size, they can be guessed in polynomial time. Next, in order to check whether the guessed generating system generates a bisimulation, we need to verify the (SYS) conditions (for $S_1 = S_2 = [1,r]$) for each of the polynomially many elements of $\mathrm{BASE}_\mathcal{G}$. Note that this will involve polynomially many membership tests for $gen(\mathcal{G})$, each of which can be performed in polynomial time by Lemma 28. If the guess leads to a non-bisimulation, we return NO. Otherwise, we use another membership test for $gen(\mathcal{G})$ to check whether the given instance of the bisimilarity problem belongs to $gen(\mathcal{G})$. We return the outcome of that test as the final result. ∎

Remark 30. Note that symbolic bisimulations are based on *partial finite permutations*, which form inverse semigroups. Consequently, inverse semigroup-theoretic structure could seem the most natural kind of structure with which to approach our problems. Unfortunately, inverse semigroups do not admit analogous results.

- There exist inverse subsemigroups of $\mathcal{IS}_n$ that require

$\binom{n}{\frac{n}{2}} \approx 2^n \sqrt{\frac{2}{\pi n}}$ generators, e.g. $\{\mathsf{id}_X | X \subseteq [1,n], |X| = \frac{n}{2}\}$.
- In the appendix we show that the membership problem for inverse subsemigroups of $\mathcal{IS}_n$ is PSPACE-complete, sharpening a result of Kozen [13].

Consequently, we were forced to look a bit deeper and base generating systems on groups.

Remark 31. Note that we do not have a matching lower bound for RA(SF), even though there are numerous NP-complete problems (or problems not known to be in PTIME) based on guessing permutations (e.g. graph/subgraph isomorphism) that might seem good candidates for a reduction. However, it turns out difficult to construct RA's of polynomial size where the induced bisimulation game would correspond to choosing permutations. This raises the intriguing prospect that there may still be scope for improvement in the RA(SF) case.

VII. Fresh-register automata with single assignment without erasure (FRA($S\#_0$))

In this section we examine the problems tackled in Sections IV-VI albeit in the general case of FRAs. We would like to apply the same techniques, aiming to produce the same upper bounds, yet the FRA setting raises significant additional challenges. Our approach for RAs relied on symbolic bisimulations and the group-theoretic structure that emanated from them. While we can express bisimilarity in FRAs symbolically following [25], we shall see that such symbolic bisimulations do not support the group-theoretic representations. The reason is the treatment of the history of the computation, which affects bisimilarity in subtle ways, especially in the initial stages of the bisimulation game. In those stages, global and local freshness can inter-simulate another, under certain conditions, which leads us to extending our symbolic representations beyond the r names that each system can have in its registers.

Simplified notation: We extend the simplified notation for RA($S\#_0$) by including transition labels for global freshness. Recall that, in any transition $q_1 \xrightarrow{t,X,i,Z} q_2$ of an r-FRA($S\#_0$), we have that $Z = \emptyset$, $i \neq 0$ and $X \in \{\circledast, \emptyset, \{i\}\}$. We thus follow a simpler notation for transitions, with $\delta \subseteq Q \times \Sigma \times \{i, i^\bullet, i^\circledast \mid i \in [1,r]\} \times Q$:

(a) we write each transition $q_1 \xrightarrow{t,\{i\},i,\emptyset} q_2$ as $q_1 \xrightarrow{t,i} q_2$;
(b) and each $q_1 \xrightarrow{t,\emptyset,i,\emptyset} q_2$ as $q_1 \xrightarrow{t,i^\bullet} q_2$;
(c) and each $q_1 \xrightarrow{t,\circledast,i,\emptyset} q_2$ as $q_1 \xrightarrow{t,i^\circledast} q_2$.

(a),(b) are as in RA($S\#_0$). In (c), the automaton reads (t,a) if a is *globally fresh*, i.e. it has not appeared in the history so far, and stores it in register i. Formally, $q \xrightarrow{t,i^\circledast} q'$ can induce a transition $(q,\rho,H) \xrightarrow{t,a} (q', \rho[i \mapsto a], H \cup \{a\})$ just if $a \notin H$.[5]

A. Symbolic bisimulation

Recall that, in the case of RAs, we were able to capture bisimilarity symbolically by using pairs of symbolic configurations of the form $((q_1,S_1),(q_2,S_2))$, whereby S_i represented $\mathsf{dom}(\rho_k)$ of the of the actual configuration (q_k,ρ_k) represented by (q_k,S_k), and a partial bijection $\sigma : S_1 \to S_2$ capturing the matching names of ρ_1 and ρ_2. Moving to FRAs, the first obstacle we face is that actual configurations contain the full history of names and have therefore unbounded size. For bisimulation purposes, though, keeping track of the whole history, or its size, is not necessary. In fact, history only plays a role in globally fresh transitions and one can easily see that the following rule:

- Every globally fresh transition from q_1 must be matched by a globally or a locally fresh transition from q_2.

is sound for simulation of globally fresh transitions.

However, global freshness leads to severe complications in the simulation of locally fresh transitions. For example, assuming configurations $(q_1,\rho_1,H),(q_2,\rho_2,H)$ with $\mathsf{rng}(\rho_1) = H$, we can see that a transition $q_1 \xrightarrow{t,1^\bullet} q_1'$ can be matched by some $q_2 \xrightarrow{t,1^\circledast} q_2'$, as the local names of q_1 coincide with all the names in H. On the other hand, if $H = \{d_1,d_2\}$ and $\rho_i = \{(1,d_i)\}$ (for $i = 1,2$), then a transition $q_1 \xrightarrow{t,1^\bullet} q_1'$ cannot be matched by some $q_2 \xrightarrow{t,1^\circledast} q_2'$ alone; rather, an additional transition $q_2 \xrightarrow{t,1} q_2''$ is needed in order to capture the fact that $q_1 \xrightarrow{t,1^\bullet} q_1'$ can produce d_2. However, if $|H| > 2r$ then there will always be some $d \in H \setminus (\mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2))$ that can be produced by $q_1 \xrightarrow{t,1^\bullet} q_1'$ and, thence, the only way for q_2 to capture it would be by some locally fresh transition.

From our discussion above it follows that, under certain circumstances which include the fact that $|H| \leq 2r$, local freshness can be captured by global freshness and some known-name transitions. To accommodate this feature, we will design symbolic bisimulations with an additional component $h \in [0,2r] \cup \{\infty\}$ that will abstract the size of $|H|$. The value $h = \infty$ would signify that $|H| > 2r$ and therefore local-fresh cannot be matched by global-fresh. On the other hand, $h \leq 2r$ would mean that $|H| = h \leq 2r$ and therefore extra care would need to be taken for fresh transitions. For $h \leq 2r$, we will consider symbolic configurations (q_i,S_i) $(i = 1,2)$ where $S_i \leq [1,3r]$ and $h = |S_i|$, related by bijections $\sigma : S_1 \to S_2$.

- The component $S_i \cap [1,r]$ of S_i will still represent the domain of ρ_i.
- The complementary part $S_i \setminus [1,r]$ will represent the remaining names, those that have passed but no longer reside in ρ_i (i.e. $H \setminus \mathsf{rng}(\rho_i)$), in some canonical fashion.

Effectively, the above will allow us to symbolically represent the history of each FRA, up to the size $2r$, in an ordered way. It will also offer us a way to decide the simulation game for locally fresh transitions. Let us say that one system performs a transition $q_1 \xrightarrow{t,i^\bullet} q_1'$:

1. Such a transition can capture any name d that is represented in some $i' \in S_1 \setminus [1,r]$. If $\sigma(i') \in [1,r]$ then the other system has the name in its registers and can (only) capture it by some $q_2 \xrightarrow{t,\sigma(i')} q_2'$.
2. If $\sigma(i') \in S_2 \setminus [1,r]$ then the name is historical and the other system does not currently have it in its registers.

[5] The latter condition above is slightly different but equivalent to that used in [25]. In *loc. cit.*, the names of ρ are not necessarily included in H and hence in this rule one stipulates that $a \notin \mathsf{rng}(\rho) \cup H$.

It is therefore obliged to simulate by some locally fresh transition $q_2 \xrightarrow{t,j^\bullet} q_2'$.

3. The transition can also capture any name d that is not in H and, in this case, the other system can capture it by any $q_2 \xrightarrow{t,j^\bullet/j^\circledast} q_2'$. Moreover, such a simulation step would increase the size of h by one.

We therefore formulate symbolic bisimulation as follows.

Definition 32. Let $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F\rangle$ be an r-FRA($S\#_0$). We first set:

$$\mathcal{U}_0 = Q \times \mathcal{P}([1,3r]) \times \mathcal{IS}_{3r} \times Q \times \mathcal{P}([1,3r]) \times ([0,2r] \cup \{\infty\})$$

$$\begin{aligned}\mathcal{U} = \{(q_1, S_1, \sigma, q_2, S_2, h) \in \mathcal{U}_0 \mid\ & \sigma \subseteq S_1 \times S_2 \\ & \wedge\, h \leq 2r \implies |\sigma| = |S_1| = |S_2| = h \\ & \wedge\, h = \infty \implies (\sigma \in \mathcal{IS}_r \wedge S_1, S_2 \subseteq [1,r])\}\end{aligned}$$

A *symbolic simulation* on $\mathcal{A}$ is a relation $R \subseteq \mathcal{U}$, with elements $(q_1, S_1, \sigma, q_2, S_2, h) \in R$ written $(q_1, S_1)\, R^h_\sigma\, (q_2, S_2)$, such that all $(q_1, S_1, \sigma, q_2, S_2, h) \in R$ satisfy the following *fresh symbolic simulation conditions* (FSYS):[6]

(a) for all $q_1 \xrightarrow{t,i} q_1'$,
   1. if $\sigma(i) \in [1,r]$ then there is some $q_2 \xrightarrow{t,\sigma(i)} q_2'$ with $(q_1', S_1)\, R^h_\sigma\, (q_2', S_2)$,
   2. if $\sigma(i) = j' \in [r{+}1, 3r]$ then there is some $q_2 \xrightarrow{t,j^\bullet} q_2'$ with $(q_1', S_1)\, R^h_{(j\,j')\circ\sigma}\, (q_2', (j\,j')\cdot S_2)$,
   3. if $i \in S_1 \setminus \mathsf{dom}(\sigma)$ then there is some $q_2 \xrightarrow{t,j^\bullet} q_2'$ with $(q_1', S_1)\, R^h_{\sigma[i\mapsto j]}\, (q_2', S_2[j])$;

(b) for all $q_1 \xrightarrow{t,i^\bullet} q_1'$, $i' \in S_1 \setminus [1,r]$ and $j \in S_2 \setminus \mathsf{rng}(\sigma)$,
   1. if $\sigma(i') \in [1,r]$ then there is some $q_2 \xrightarrow{t,\sigma(i')} q_2'$ with $(q_1', (i\,i')\cdot S_1)\, R^h_{\sigma\circ(i\,i')}\, (q_2', S_2)$,
   2. if $\sigma(i') = j' \in [r{+}1, 3r]$ then there is some $q_2 \xrightarrow{t,j^\bullet} q_2'$ with $(q_1', (i\,i')\cdot S_1)\, R^h_{(j\,j')\circ\sigma\circ(i\,i')}\, (q_2', (j\,j')\cdot S_2)$,
   3. there exists $q_2 \xrightarrow{t,j} q_2'$ with $(q_1', S_1[i])\, R^h_{\sigma[i\mapsto j]}\, (q_2', S_2)$;

(c) for all $q_1 \xrightarrow{t,\ell_i} q_1'$ with $\ell_i \in \{i^\bullet, i^\circledast\}$ there is some $q_2 \xrightarrow{t,\ell_j} q_2'$ with $\ell_j \in \{j^\bullet, j^\circledast\}$ and,
   1. if $h < 2r$ then, taking $i' = \min([r{+}1, 3r] \setminus S_1)$ and $j' = \min([r{+}1, 3r] \setminus S_2)$, we have $(q_1', (i\,i')\cdot S_1[i'])\, R^{h+1}_{(i\,i')\circ\sigma[i'\mapsto j']\circ(j\,j')}\, (q_2', (j\,j')\cdot S_2[j'])$;
   2. if $h = 2r$ then $(q_1', S_1[i] \cap [1,r])\, R^\infty_{\sigma[i\mapsto j]\cap[1,r]^2}\, (q_2', S_2[j] \cap [1,r])$;
   3. if $h = \infty$ then $(q_1', S_1[i])\, R^\infty_{\sigma[i\mapsto j]}\, (q_2', S_2[j])$ and if $\ell_i = i^\bullet$ then $\ell_j = j^\bullet$.

We let the inverse of R be

$$R^{-1} = \{\, (q_2, S_2, \sigma^{-1}, q_1, S_1) \mid (q_1, S_1, \sigma, q_2, S_2) \in R \,\}$$

and call R a ***symbolic bisimulation*** if both R and R^{-1} are symbolic simulations. We let *s-bisimilarity*, denoted $\overset{\mathsf{s}}{\sim}$, be the union of all symbolic bisimulations.

As before, we define a sequence of ***indexed bisimilarity*** relations $\overset{i}{\sim} \subseteq \mathcal{U}$ inductively as follows. We let $\overset{0}{\sim}$ be the whole of $\mathcal{U}$. Then, for all $i \in \omega$ and $h \in [0,2r] \cup \{\infty\}$, $(q_1, S_1) \overset{i+1}{\sim}{}^h_\tau (q_2, S_2)$ just if both $(q_1, S_1, \tau, q_2, S_2, h)$ and $(q_2, S_2, \tau^{-1}, q_1, S_1, h)$ satisfy the (FSYS) conditions in $\overset{i}{\sim}$.

Let $\kappa_i = (q_i, \rho_i, H)$ $(i = 1, 2)$ be configurations with common history H and let $n = |H|$. Their symbolic representation will depend on n. We take $\mathsf{symb}(\kappa_1, \kappa_2) \subseteq \mathcal{U}$ to be:

$$\begin{cases} \{(q_1, \mathsf{dom}(\hat\rho_1), \hat\rho_1; \hat\rho_2^{-1}, q_2, \mathsf{dom}(\hat\rho_2), n) \mid \theta(\hat\rho_1, \hat\rho_2)\} & n \leq 2r \\ \{(q_1, \mathsf{dom}(\rho_1), \rho_1; \rho_2^{-1}, q_2, \mathsf{dom}(\rho_2), \infty)\} & n > 2r \end{cases}$$

where $\theta(\hat\rho_1, \hat\rho_2)$ is the condition stipulating that $\hat\rho_i$ range over all $3r$-register assignments of type $S\#_0$ such that $\mathsf{rng}(\hat\rho_i) = H$ and $\hat\rho_i \restriction [1,r] = \rho_i$, for $i = 1, 2$. In particular, $\mathsf{symb}(\kappa_1, \kappa_2)$ is singleton in case $n > 2r$ but not necessarily so if $n \leq 2r$. The following lemma ensures that, with respect to bisimilarity, the specific choice of element from $\mathsf{symb}(\kappa_1, \kappa_2)$ is not important.

Lemma 33. *For all κ_1, κ_2 as above, if $|H| < 2r$ then either $\mathsf{symb}(\kappa_1, \kappa_2) \subseteq \overset{\mathsf{s}}{\sim}$ or $\mathsf{symb}(\kappa_1, \kappa_2) \cap \overset{\mathsf{s}}{\sim} = \emptyset$.*

Definition 34. We say that κ_1 and κ_2 are *s-bisimilar*, written $\kappa_1 \overset{\mathsf{s}}{\sim} \kappa_2$, if $\mathsf{symb}(\kappa_1, \kappa_2) \subseteq \overset{\mathsf{s}}{\sim}$.

Note how the (FSYS) conditions are divided with respect to the value of h: conditions (a2), (b1), (b2), (c1) and (c2) all require $h \leq 2r$; while conditions (a3), (b3) and (c3) are for $h = \infty$. On the other hand, (a1) applies to all h.

Remark 35. The definition of symbolic bisimulation we give here is crucially more fine-grained than the one in [25]. Although in *loc. cit.* the symbolic bisimulation is also given parametrically to the size of the history h (up to the given bound[7]), for $h \leq 2r$ that formulation is simplistic in that it only keeps track of names that reside in registers of the automata,[8] which in turn prohibits us to derive $(q_1, S_1)\, R^h_{\sigma_1;\sigma_2}\, (q_3, S_3)$ from $(q_1, S_1)\, R^h_{\sigma_1}\, (q_2, S_2)$ and $(q_2, S_2)\, R^h_{\sigma_2}\, (q_3, S_3)$ and apply the group-theoretic approach.

Lemma 36. *Let κ_1 and κ_2 be configurations of an r-FRA($S\#_0$), then: $\kappa_1 \sim \kappa_2 \iff \kappa_1 \overset{\mathsf{s}}{\sim} \kappa_2$. Moreover, for all $i \in \omega$, $\overset{i+1}{\sim} \subseteq \overset{i}{\sim}$ and $(\bigcap_{i\in\omega} \overset{i}{\sim}) = \overset{\mathsf{s}}{\sim}$.*

Similarly to symbolic bisimulations for RA($S\#_0$), we have the following closure properties. Given $R \subseteq \mathcal{U}$ we split R into *components*:

$$R = \sum\nolimits_{h\in[0,2r]\cup\{\infty\}} R^h$$

where $R^h = \{(q_1, S_1, \sigma, q_2, S_2) \mid (q_1, S_1, \sigma, q_2, S_2, h) \in R\}$. We now write $Cl(R)$ for the componentwise closure of R with respect to identity, symmetry, transitivity and extension of partial permutations, i.e. $Cl(R) = \sum_{h\in[0,2r]\cup\{\infty\}} Cl(R^h)$.

Proposition 37. *Symbolic bisimilarity and indexed symbolic bisimilarity for FRA($S\#_0$) are closed.*
1) $Cl(\overset{\mathsf{s}}{\sim}) = \overset{\mathsf{s}}{\sim}$; 2) for all $i \in \omega$: $\overset{i}{\sim} = Cl(\overset{i}{\sim})$.

[6] We say that $(q_1, S_1, \sigma, q_2, S_2, h)$ *satisfies the* (FSYS) *conditions in R.*

[7] In fact, the bound used in [25] is smaller $(2r{-}1)$, due to the fact that it examines bisimulation between configurations with common initial names.

[8] that is, in $(q_1, S_1)\, R^h_\sigma\, (q_2, S_2)$ we always have $S_1, S_2 \subseteq [1,r]$.

We therefore observe that the extension of symbolic representations to the size $3r$, and the ensuing history representation up to size $2r$ along with the extended symbolic bisimulation conditions, have paid off in yielding the desired closure properties. The group-theoretic behaviour of a closed relation R differs between different components:

- R^∞ has the same structure as the closed relations R examined in Section IV-B.
- For $h \in [0, 2r]$, the tuples $(q_1, S_1, \sigma, q_2, S_2) \in R^h$ respect the condition $|S_1| = |S_2| = |\sigma| = h$. In particular, σ is a bijection from S_1 to S_2 and, hence, in this case closure under extension is trivial, and so are characteristic sets ($X^p_S(R^h) = S$). Moreover, $\sigma \in \mathcal{IS}_{3r}$ and $S_1, S_2 \subseteq [1, 3r]$.

We can hence see that the same groups arise as in the case of RA($S\#_0$), and actually simpler in the case $h \in [0, 2r]$, albeit parameterised over h. This allows for a similar group-theoretic treatment.

B. PSPACE bound for bisimulation game

Lemma 38. *Let $h \in [0, 2r] \cup \{\infty\}$, $S_1, S_2 \subseteq [1, 3r]$ and $\mathcal{U}^h_{S_1,S_2} = Q \times \{S_1, S_2\} \times \mathcal{IS}_r \times Q \times \{S_1, S_2\} \times \{h\}$. Then the sub-chain $\{\overset{i}{\sim} \mid (\overset{i+1_h}{\sim} \cap\, \mathcal{U}^h_{S_1,S_2}) \subsetneq (\overset{i_h}{\sim} \cap\, \mathcal{U}^h_{S_1,S_2})\}$ has size $O(|Q|^2 + r^2|Q|)$.*

Given $S_1, S_2 \subseteq [1, 3r]$ and $h \in [0, 2r] \cup \{\infty\}$, let us call the triple (S_1, S_2, h) ***proper*** just if: either $|S_1| = |S_2| = h$, or $h = \infty$ and $S_1, S_2 \subseteq [1, r]$. For such (S_1, S_2, h), let us define:

$$\hat{\gamma}(S_1, S_2, h) = \begin{cases} \gamma(S_1 \cap [1, r], S_2 \cap [1, r]) + h & \text{if } h \in [0, 2r] \\ \gamma(S_1, S_2) + 2r + 1 & \text{if } h = \infty \end{cases}$$

The measure $\hat{\gamma}$ enables us to show the following bound for stabilising indexed bisimulation, proven similarly to Lemma 19.

Lemma 39. *Let $\mathcal{U}^{h-}_{S_1,S_2} = Q \times \{S_1\} \times \mathcal{IS}_r \times Q \times \{S_2\} \times \{h\}$ and let $\hat{c}$ be the constant of $O(|Q|^2 + r^2|Q|)$ in Lemma 38 (2).*

1) *For any proper (S_1, S_2, h), we have $(\overset{j}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2}) = (\overset{\mathsf{s}}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2})$, where $j = \hat{c}(4r - \hat{\gamma}(S_1, S_2, h) + 2)(|Q|^2 + r^2|Q|)$.*
2) *Let $B = \hat{c}(4r + 2)(|Q|^2 + r^2|Q|)$. For any proper (S_1, S_2, h), it holds that $\overset{B}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2} = \overset{\mathsf{s}}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2}$.*

We can therefore establish PSPACE solvability.

Proposition 40. *For any FRA($S\#_0$) bisimulation problem, if there is a winning strategy for Attacker then there is one of depth $O(r|Q|^2 + r^3|Q|)$.*

Proposition 41. *$\sim$-FRA($S\#_0$) is solvable in PSPACE.*

C. Generating systems and NP routines

We proceed to generating systems for FRA(SF), which are h-parameterised versions of the ones for RA(SF) expanding over $[1, 3r]$. Since we again consider only the closed relation $\overset{\mathsf{s}}{\sim}$, we will omit this argument to characteristic sets and groups. We call a pair (S, h) proper just if (S, S, h) is proper.

Definition 42. A ***generating system*** $\mathcal{G}_{S,h}$ for proper (S, h) (in which case $|S| \leq 2r$), consists of:

- a partitioning of Q into $P_1, \cdots, P_k$;
- for each partition P_i, a single representative $p_i \in P_i$ and:
  - a characteristic set $X^{p_i}_{S,h} \subseteq S$;
  - a set $G^{p_i}_{S,h}$, of up to $\max(2,r)$ permutations $\sigma \in \mathcal{S}_{X^{p_i}_{S,h}}$;
  - for each $q \in P_i \setminus \{p_i\}$, a partial permutation $\mathsf{ray}^{p_i}_q \in \mathcal{IS}_S$ such that $\mathsf{dom}(\mathsf{ray}^{p_i}_q) = X^{p_i}_{S,h}$; for technical convenience, we also add $\mathsf{ray}^{p_i}_{p_i} = \mathsf{id}_{X^{p_i}_{S,h}}$.

We write $rep(\mathcal{G}_{S,h})$ for the set $\{p_1, \cdots, p_k\}$ of representatives. From $\mathcal{G}_{S,h}$ we generate $gen(\mathcal{G}_{S,h}) \subseteq (Q \times \{S\} \times \mathcal{IS}_{3r} \times Q \times \{S\})$ by setting

$$\begin{aligned} \mathrm{BASE}_{\mathcal{G}_{S,h}} = &\{(p_i, S, \sigma, p_i, S) \mid p_i \in rep(\mathcal{G}_{S,h}) \wedge \sigma \in G^{p_i}_{S,h}\} \\ &\cup \{(p_i, S, \mathsf{ray}^{p_i}_q, q, S) \mid p_i \in rep(\mathcal{G}_{S,h}) \wedge q \in P_i\} \end{aligned}$$

and taking $gen(\mathcal{G}_{S,h}) = \mathit{Cl}(\mathrm{BASE}_{\mathcal{G}_{S,h}})$.

The following lemma, proved in the same way as Lemmata 27 and 28, enables us to prove an NP upper bound for bisimilarity in FRA(SF).

Lemma 43. *1) For any proper (S, h) there exists a generating system $\mathcal{G}_{S,h}$ such that $gen(\mathcal{G}_{S,h}) = \overset{\mathsf{s}}{\sim} \cap\, \mathcal{U}^h_{S,S}$.*
2) For any generating system $\mathcal{G}_{S,h}$, membership in $gen(\mathcal{G}_{S,h})$ can be determined in polynomial time.

Theorem 44. *$\sim$-FRA(SF) is solvable in NP.*

Proof: Given an input tuple $(q_1, S_1, \sigma, q_2, S_2, h^0)$, note first that $[1, r] \subseteq S_1, S_2$ (by F) and $|S_1| = |S_2|$. We can therefore convert it to an equivalent $(q_1, S'_1, \sigma', q_2, S_2, h^0)$, with $S'_1 = S_2$, by applying a permutation on the indices in $S_1 \setminus [1, r]$. Hence, we can assume wlog that our input is some $(q_1, S^0, \sigma, q_2, S^0, h^0)$. Moreover, because the expansion of S in the symbolic bisimulation game (when $h \in [0, 2r]$) always occurs in its first free register ($\min([r+1, 3r] \setminus S)$), we can compute the sequence $(S^0, h^0, S^0), (S^1, h^0{+}1, S^1), \cdots$ of distinct triples considered in the game (in the $h \in [0, 2r]$ phase), which must thence be bounded in length by $2r$. Including the final bisimulation phase ($h = \infty$), this gives us $2r + 1$ phases. We first generate for each of them a generating system, say $\mathcal{G}_{S^i,h^i}$, and then verify whether each $gen(\mathcal{G}_{S^i,h^i})$ is a symbolic bisimulation, similarly to Theorem 29. Note that each such check can be achieved in polynomial time. If the guess leads to some $gen(\mathcal{G}_{S^i,h^i})$ being a non-symbolic-bisimulation, we return NO. Otherwise, we use another membership test for $gen(\mathcal{G}_{S^0,h^0})$ to check whether the given instance of the bisimilarity problem belongs to $gen(\mathcal{G}_{S^0,h^0})$. We return the outcome of that test as the final result. ∎

VIII. Visibly pushdown automata with single assignment and filled registers (VPDRA(SF))

Finally, we consider a variant of register automata with visible pushdown storage [2]. We only consider the most restrictive register discipline (SF), as undecidability will be shown to apply already in this case.

Definition 45. A ***visibly pushdown r-register automaton*** (r-VPDRA(SF)) $\mathcal{A}$ is a tuple $\langle Q, \Sigma_C, \Sigma_N, \Sigma_R, \Gamma, \rho_I, \delta \rangle$, where Q, ρ_I have the same meaning as for r-RA,

- Σ_C, Σ_N, Σ_R are disjoint finite sets of *push-*, *no-op-* and *pop-tags* respectively;
- Γ is a finite set of *stack tags*;
- $\delta = \delta_C \cup \delta_N \cup \delta_R$, the transitions, have $Lab = \{1, \ldots, r\} \cup \{1^\bullet, \ldots, r^\bullet\}$ and:
  - $\delta_C \subseteq Q \times \Sigma_C \times Lab \times \Gamma \times \{1, \cdots, r\} \times Q$
  - $\delta_N \subseteq Q \times \Sigma_N \times Lab \times Q$
  - $\delta_R \subseteq Q \times \Sigma_R \times Lab \times \Gamma \times \{1, \cdots, r, \bullet\} \times Q$

Configurations of r-VPDRA(SF) are triples (q, ρ, s), where $q \in Q$, ρ is a register assignment and $s \in (\Gamma \times \mathcal{D})^*$ is the stack. An LTS arises by having a labelled edge $(q_1, \rho_1, s_1) \xrightarrow{(t,d)} (q_2, \rho_2, s_2)$ just if there exist $i \in [1, r]$ and $l \in \{i, i^\bullet\}$ such that: (i) $\rho_1(x) = \rho_2(x)$ for all $x \neq i$; (ii) if $l = i$ then $\rho_1(i) = \rho_2(i)$, otherwise $\rho_2(i) \notin \mathsf{rng}(\rho_1)$; and (iii) one of the following conditions holds:

- $(q_1, t, l, t', j, q_2) \in \delta_C$ and $s_2 = (t', \rho_2(j))s_1$,
- $(q_1, t, l, q_2) \in \delta_N$ and $s_2 = s_1$,
- $(q_1, t, l, t', j, q_2) \in \delta_R$, $s_1 = (t', d')s_2$,

where if $j \in [1, r]$ then $d' = \rho_2(j)$, otherwise $d' \notin \mathsf{rng}(\rho_2)$.

We show that, in the infinite-alphabet setting, even the visibly pushdown case is undecidable and even under the most stringent register discipline. To do so, we reduce from the undecidable emptiness problem for (one-way) *universal* register automata with two registers (URA_2) [7]. URA_2 are equipped with two registers and an input tape, which moves forward after an explicit instruction. While the head is scanning the current input symbol, the automaton can compare it with its register content and branch accordingly. Consequently, an input symbol can be read and processed without being stored in registers.

Theorem 46. *VPDRA(SF) bisimiliarity is undecidable.*

Proof (sketch): Given a URA_2 U, we devise a 2-VPDRA $\mathcal{A}_U$ with two configurations κ_1, κ_2 such that U accepts a word iff $\kappa_1 \not\sim \kappa_2$. $\mathcal{A}_U$ is constructed to induce a bisimulation game in which Attacker gets a chance to choose a word to be accepted by U and simulate an accepting run (if one exists). The stack of $\mathcal{A}_U$ is used to store the word that Attacker has chosen, with the top of the stack playing the role of the head of U and the two registers of $\mathcal{A}_U$ emulating the two registers of U. To simulate a transition we arrange for Attacker to guess the outcome of the comparison of the top of stack with the current register contents whilst allowing Defender to verify the correctness of such guesses via Defender forcing. Transitions from universal states are chosen by Defender, again using Defender forcing. ∎

The argument sketched above also reduces URA_1 emptiness to 1-VPDRA, which implies a non-primitive-recursive lower bound for 1-VPDRA.

IX. Conclusion

We have demonstrated bounds on the bisimilarity problem for broad classes of register and fresh-register automata, including those studied in the literature. We have shown that, on the one hand, the ability to start with empty registers, the ability to erase the contents of registers (or equivalently, store duplicate values) and the addition of a stack all affect the inherent complexity of bisimulation testing. Whilst, on the other, global freshness does not seem to affect complexity. Except in the case of automata in which registers are required to be filled, all our bounds are tight.

Finally, we note that although we have formulated the bisimulation problems with respect to two configurations of a single automaton, extending our results to a formulation of problems concerning two automata is unproblematic. If the automata have different numbers of registers, then the bisimulation game can be played over an automaton with a number of registers equal to the maximum of the two constituent systems, with additional registers initialised (and left) empty. Even in F register disciplines, it follows from our arguments that, since these extra registers are never assigned to, the composite system can be treated as a $\#_0$ system without compromising on complexity.

References

[1] R. Alur, P. Cerný, and S. Weinstein. Algorithmic analysis of array-accessing programs. In *CSL*, *LNCS*, pp. 86–101. Springer, 2009.
[2] R. Alur and P. Madhusudan. Visibly pushdown languages. In *STOC*, pp. 202–211, 2004.
[3] M. F. Atig, A. Bouajjani, and S. Qadeer. Context-bounded analysis for concurrent programs with dynamic creation of threads. *Log. Meth. in Comput. Sci.*, 7(4), 2011.
[4] L. Babai. On the length of subgroup chains in the symmetric group. *Com. in Algebra*, 14(9), 1986.
[5] M. Benedikt, S. Göller, S. Kiefer, and A. S. Murawski. Bisimilarity of pushdown automata is nonelementary. In *LICS*, pp. 488–498. IEEE Computer Society, 2013.
[6] M. Boreale and L. Trevisan. A complexity analysis of bisimilarity for value-passing processes. *Theor. Comput. Sci.*, 238(1-2):313–345, 2000.
[7] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009.
[8] M. L. Furst, J. E. Hopcroft, and E. M. Luks. Polynomial-time algorithms for permutation groups. In *FOCS*, pp. 36–41. IEEE Computer Society, 1980.
[9] R. Grigore, D. Distefano, R. L. Petersen, and N. Tzevelekos. Runtime verification based on register automata. In *TACAS*, *LNCS*, pp. 260–276. Springer, 2013.
[10] P. Jančar and J. Srba. Undecidability of bisimilarity by defender's forcing. *J.ACM*, 55(1), 2008.
[11] B. Jonsson and J. Parrow. Deciding bisimulation equivalences for a class of non-finite-state programs. *Inf. Comput.*, 107(2):272–302, 1993.
[12] M. Kaminski and N. Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2), 1994.
[13] D. Kozen. Lower bounds for natural proof systems. In *FOCS*, pp. 254–266. IEEE, 1977.
[14] A. McIver and P. M. Neumann. Enumerating finite groups. *Quart. J. Math.*, 38(4), 1987.
[15] U. Montanari and M. Pistore. An introduction to history dependent automata. *Electr. Notes Theor. Comput. Sci.*, 10, 1997.
[16] A. S. Murawski, S. J. Ramsay, and N. Tzevelekos. Reachability in pushdown register automata. In *MFCS*, *LNCS*, pp. 464–473. Springer, 2014.
[17] F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004.
[18] M. Pistore. *History Dependent Automata*. PhD thesis, University of Pisa, 1999.
[19] H. Sakamoto. *Studies on the Learnability of Formal Languages via Queries*. PhD thesis, Kyushu University, 1998.
[20] H. Sakamoto and D. Ikeda. Intractability of decision problems for finite-memory automata. *Theor. Comput. Sci.*, 231(2):297–308, 2000.
[21] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL*, *LNCS*, pp. 41–57. Springer, 2006.
[22] G. Sénizergues. The bisimulation problem for equational graphs of finite out-degree. *SIAM J. Comput.*, 34(5):1025–1106, 2005.
[23] J. Srba. Visibly pushdown automata: From language equivalence to simulation and bisimulation. In *CSL*, *LNCS*, pp. 89–103. Springer, 2006.
[24] J. Srba. Roadmap of infinite results.http://www.brics.dk/~srba/roadmap/, 2008.
[25] N. Tzevelekos. Fresh-register automata. In *POPL*, pp. 295–306. ACM Press, 2011.

APPENDIX A
PROOFS FROM SECTION II

A. Proof of Lemma 6

Proof: First note that, for all XY, any RA(XY) $\mathcal{A}$ can be trivially seen as an FRA(XY) $\mathcal{A}'$ (i.e. $\mathcal{A}'$ has the same components as $\mathcal{A}$). We claim that, for any pair $(q_1, \rho_1), (q_2, \rho_2)$ of RA-configurations of $\mathcal{A}$,

$$(q_1, \rho_1) \sim (q_2, \rho_2) \iff (q_1, \rho_1, H) \sim (q_2, \rho_2, H) \qquad (*)$$

where $H = \mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2)$ and $(q_1, \rho_1, H), (q_2, \rho_2, H)$ are configurations of $\mathcal{A}'$. Indeed, we can show that the relation between $\mathcal{A}$- and $\mathcal{A}'$-configurations given by:

$$R = \{\ ((q, \rho), (q, \rho, H)) \mid \mathsf{rng}(\rho) \subseteq H\ \}$$

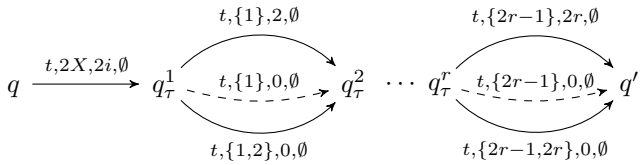
is a bisimulation, from which we obtain $(*)$.

We next show the FRA-bisimilarity inclusions; the RA-bisimilarity inclusions are shown in a similar (simpler) way.

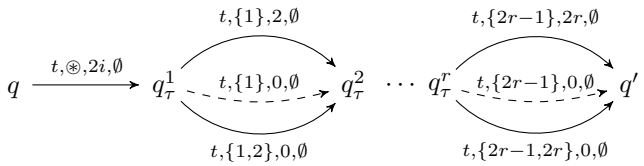
Observe that, for any $X \in \{S, M\}$, $\sim$-FRA(XF) $\leq$ $\sim$-FRA($X\#_0$) $\leq$ $\sim$-FRA($X\#$). This is because any FRA(XF) can be viewed trivially as an FRA($X\#_0$) in which all registers begin filled and, similarly, any FRA($X\#_0$) can be viewed trivially as an FRA($X\#$) in which no registers are ever erased.

Now, given an r-FRA($S\#$) $\mathcal{A}$ and two configurations κ_1 and κ_2 we construct a $2r$-FRA(MF) $\mathcal{A}'$ and configurations $\widehat{\kappa_1}$ and $\widehat{\kappa_2}$ in which every register k of $\mathcal{A}$ is simulated by two registers $2k-1$ and $2k$ of $\mathcal{A}'$. The states of $\mathcal{A}'$ are the states of $\mathcal{A}$ augmented by an additional state q^i_τ for every $q \in Q$, $i \in [1, r]$ and every $\tau \in \delta$. The representation scheme is as follows: if registers $2k-1$ and $2k$ of $\mathcal{A}'$ contain the same letter then register k of $\mathcal{A}$ is empty, otherwise the register k in $\mathcal{A}$ contains exactly the contents of register $2k$ in $\mathcal{A}'$.

Each transition $\tau = q \xrightarrow{t,X,i,Z} q'$ of $\mathcal{A}$, in which $X \subseteq [1, r]$, is simulated by a sequence of transitions of $\mathcal{A}'$ with the following shape:



where $2X$ is a shorthand for $\{2x \mid x \in X\}$ and, for each $j \in [1, r]$ the solid arrows labelled $(t, \{2j-1\}, 2i, \emptyset)$ and $(t, \{2j-1, 2j\}, 0, \emptyset)$ respectively exist just if $j \in Z$ and the dashed arrow labelled $(t, \{2j-1\}, 0, \emptyset)$ exists just if $j \notin Z$. On the other hand, each transition $\tau = q \xrightarrow{t,\circledast,i,Z} q'$ of $\mathcal{A}$ is simulated by a sequence of transitions of $\mathcal{A}'$:



where solid and dashed arrows are as above.

We say that a pair of configurations $(q_1, \widehat{\rho_1})$, $(q_2, \widehat{\rho_2})$ of $\mathcal{A}'$ *represents* a pair of configurations (q_1, ρ_1), (q_2, ρ_2) of $\mathcal{A}$ just if $\widehat{\rho_1}$ is a representation of ρ_1 and $\widehat{\rho_2}$ is a representation of $\widehat{\rho_2}$ as discussed above and, furthermore:

- for all $k \in [1, r]$, $i \in [1, 2r]$, $j \in \{1, 2\}$: if $\widehat{\rho_j}(2k-1) = \widehat{\rho_j}(i)$ then $i \in \{2k-1, 2k\}$
- for all $k \in [1, r]$: $\widehat{\rho_1}(2k-1) = \widehat{\rho_2}(2k-1)$

These latter two properties can easily be seen to be an invariant of configurations reachable from any pair that initially satisfy it, since transitions of $\mathcal{A}'$ only write to even numbered registers $2k$ and then only then with a fresh letter or the contents of the adjacent register $2k-1$.

By construction, the automaton $\mathcal{A}'$ faithfully simulates the original in the following sense, given configurations (q_1, ρ_1), (q_2, ρ_2) of $\mathcal{A}$ and $\mathcal{A}'$ representations $\widehat{\rho_1}$ of ρ_1 and $\widehat{\rho_2}$ of ρ_2: $(q_1, \rho_1) \sim (q_2, \rho_2)$ in $\mathcal{S}(\mathcal{A})$ iff $(q_1, \widehat{\rho_1}) \sim (q_2, \widehat{\rho_2})$ in $\mathcal{S}(\mathcal{A}')$. ∎

APPENDIX B
PROOFS FROM SECTION III

A. Reduction from the bisimilarity problem for FRA(M#) to the bisimilarity problem for finite state automata

Given an instance $\langle \mathcal{A}, \kappa_1, \kappa_2 \rangle$ of the bisimilarity problem for FRA(M#), where $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F \rangle$ has r-registers, and we construct an instance $\langle \mathcal{B}, \gamma_1, \gamma_2 \rangle$ of the bisimilarity problem for finite state automata (FSA).

The idea of the construction is to induce a bisimulation game over a *finite alphabet* which simulates the original, turn by turn. To simulate a turn (Attacker move then Defender move) of the original game, the game induced by $\mathcal{B}$ will use several of its own turns. The key insight of the construction is that $2r+2$ letters of $\mathcal{D}$ are all that are required to simulate the behaviour of $\mathcal{A}$ over the whole $\mathcal{D}$, even in the presence of global freshness.

Hence, first we choose any subset $N \subseteq \mathcal{D}$ of letters of size $2r+2$.

Definition 47. Let the r-register assignments whose range is a subset of N be called *N-assignments*. An *N-configuration* is a pair (q, ρ) consisting of a state q and N-assignment ρ. We shall use the metavariable γ to refer to N-configurations. A *potted history* is a finite subset of N of size at most $2r+1$.

To help describe the construction we will use the following two definitions. The first will allow us to relate configurations of an $\mathcal{S}(\mathcal{A})$-bisimulation game to configurations of the more abstract $\mathcal{B}$-game.

Definition 48. Given two sets of letters $X_1, X_2 \subseteq \mathcal{D}$, an *$N$-representation* for (X_1, X_2) is a surjective function $\phi : \mathcal{D} \to N$ satisfying the following constraints:

(R1) $\phi \restriction X_1 : X_1 \to \phi(X_1)$ is a bijection;
(R2) for all $i \in \{1, 2\}$, $a \in \mathcal{D}$: $\phi(a) \in \phi(X_i)$ implies $a \in X_i$.

Thus, in each case, X_i partitions ϕ as $\phi = \phi^1_i \uplus \phi^2_i$ with $\mathsf{dom}(\phi^1_i) = X_i$ and $\mathsf{dom}(\phi^1_i) \cap \mathsf{dom}(\phi^2_i) = \mathsf{rng}(\phi^1_i) \cap \mathsf{rng}(\phi^2_i) = \emptyset$. In addition, ϕ^1_1 is a bijection.

We extend the action of a representation to register assignments, histories and configurations by writing:

$$(\phi \cdot \rho)(i) = \begin{cases} \# & \text{if } \rho(i) = \# \\ \phi(\rho(i)) & \text{otherwise} \end{cases}$$

$$\phi \cdot H = \{\phi(d) \mid d \in H\}$$

$$\phi \cdot (q, \rho, H) = (q, \phi \cdot \rho, \phi \cdot H)$$

If the concrete $\mathcal{A}$-game configuration is related to the abstract $\mathcal{B}$ one by an N-representation then this ensures that all the choices available to the player of the $\mathcal{A}$-game are essentially available to the player of the $\mathcal{B}$-game.

Lemma 49. *Let ρ be a register assignment, H a history containing $\mathsf{rng}(\rho)$, and $a \in \mathcal{D}$. If ϕ is an N-representation of $(\mathsf{rng}(\rho), H)$ then consider the following transitions induced by $q \xrightarrow{t,X,i,Z} q'$:*

$$(q_1, \rho, H) \xrightarrow{t,a} (q_1', \rho'[i \mapsto a], H \cup \{a\})$$
implies
$$(q_1, \phi\cdot\rho, \phi\cdot H) \xrightarrow{t,\phi(a)} (q_1', (\phi\cdot\rho')[i \mapsto \phi(a)], (\phi\cdot H)\cup\{\phi(a)\})$$

and, for $a \in N$:

$$(q_1, \phi \cdot \rho, \phi \cdot H) \xrightarrow{t,a} (q_1', (\phi \cdot \rho')[i \mapsto a], (\phi \cdot H) \cup \{a\})$$
implies, for all $b \in \phi^{-1}(a)$,
$$(q_1, \rho, H) \xrightarrow{t,b} (q_1', \rho'[i \mapsto b], H \cup \{b\})$$

where, for all $j \in [1, r]$, $\rho'(j) = \#$ if $j \in Z$ and $\rho'(j) = \rho(j)$ otherwise.

Proof: We deal with the two facts separately. For the first, we verify the validity of the concluded transition by cases. If $X = \{j \mid \rho(j) = a\}$ then by (R1,R2) also $X = \{j \mid \phi \cdot \rho(j) = \phi(a)\}$. Otherwise $X = \circledast$ and $a \notin H$, by (R2) also $\phi(a) \notin \phi \cdot H$.

For the second, let $b \in \phi^{-1}(a)$, we verify the validity of the concluded transition by cases. If $X = \{j \mid (\phi \cdot \rho)(j) = a\}$ then it follows from (R1,R2) that $X = \{j \mid \rho(j) = b\}$. If $X = \circledast$ then $a \notin \phi \cdot H$ and hence, by (R2), $b \notin H$. ∎

We note the following two facts:

Proposition 50. *Let ρ_1, ρ_2 be register assignments and $H \supseteq \mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2)$ a history.*

(i) There exists some N-representation ϕ for $(\mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2), H)$.
(ii) If ϕ is an N-representation for $(\mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2), H)$ then ϕ is an N-representation for $(\mathsf{rng}(\rho_1), H)$ and $(\mathsf{rng}(\rho_2), H)$.

Proof:

(i) Since $\mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2) \leq 2r \leq |N|$, start by choosing any injection f_1 from $\mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2)$ into N. Next, we split by cases on whether $H \subseteq \mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2)$ holds to define

$$f_2 : (H \setminus \mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2)) \to N.$$

If the condition is true then set f_2 to be $\emptyset$. Otherwise choose any $d \in N \setminus f_1(\mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2))$ and define the constant function f_2 by $f_2(\ell) = d$ for all $\ell \in H \setminus \mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2)$. Finally, because N has size $2r + 2$, $N \setminus (f_1 \cup f_2)(H) \neq \emptyset$ and hence one may obtain some surjection

$$f_3 : (\mathcal{D} \setminus H) \to (N \setminus (f_1 \cup f_2)(H))$$

The required representation is then $f_1 \cup f_2 \cup f_3$.

(ii) Immediate from the definition.

∎

What makes the reduction difficult is that, having chosen a representation ϕ, the $\mathcal{B}$-configuration arrived at after simulating an $\mathcal{A}$-transition may no longer be correctly represented by ϕ. This is because, due to the restriction on the size of N, it is necessary to "recycle" names from the new history $\phi \cdot (H \cup \{a\})$, since the new history may become identified with N and hence there are no spare letters with which to faithfully simulate globally fresh transitions.

So, if we say that a and b are two letters from N, it can be that in one transition a is a letter local to the current configuration and b is played as a globally fresh one and stored in place of a in the registers. We expect that afterwards both of a and b are in the history, preventing them from being played as globally fresh letters, but if a no longer appears in the registers it may need to be recycled by removing it from the subsequent history.

Hence, after simulating a transition, $\mathcal{B}$ must discard the history it has reached in favour of a new (but still related one) in which it has recycled some names. We say that the new history is a potting of the old one and the next definition allows to describe the valid class of potted histories for a given (concrete) history and a distinguished subset.

Definition 51. Given histories H and $\hat{H}$ and set of letters $X \subseteq H$ such that $|X| \leq 2r$, $\hat{H}$ is a *potting of H wrt X* whenever:

(H1) $X \subseteq \hat{H}$
(H2) $|\hat{H}| = \lceil |H| \rceil^{2r+1}$.

As we shall see, the set X will always be taken to be the range of the two register assignments of the current position in the game. This will ensure that the abstract, potted history always contains all the letters in the registers of the current game position and hence it will be impossible for a player of the $\mathcal{B}$-game to make a globally fresh transition using a letter which is actually already contained in a register of the opposite system. However, for the potted history to be chosen in this way requires that the player who chooses it know the contents of both of the register assignments of the current position of the game. For this reason, we design $\mathcal{B}$ so that a state contains *both* the (representations of the) register assignments of the

two configurations involved in the current position of the $\mathcal{A}$-game which is being simulated.

Let us start to unpack this by making an artificial distinction between the two "systems" involved in the bisimulation game. Recall that a configuration of the bisimulation game is a pair (κ, κ') of configurations of $\mathcal{A}$ where κ has been reached from κ_1 and κ' has been reached from κ_2 according to the rules of the game. We shall refer to the left component of this pair as the *left system* and the right component as the *right system*. We aim to design $\mathcal{B}$ so as to simulate the bisimulation game induced by $\mathcal{A}$ and, to this end, the left systems in the $\mathcal{B}$ will represent the left system in the $\mathcal{A}$ game and conversely. However, for the reasons given above a configuration of $\mathcal{B}$ (which is just a state of $\mathcal{B}$) contains a snapshot of a configuration of the $\mathcal{A}$ bisimulation game (i.e. components from both the left and right system configurations), plus some additional bookeeping information. Then, to tell apart the two $\mathcal{B}$ systems, we tag configurations with either (L)eft or (R)ight.

We make the construction of $\mathcal{B}$ precise as follows. We build the alphabet as (note that δ is the finite transition relation of $\mathcal{A}$):

$$\Sigma = (\delta \times N) \cup (\delta \times N \times \mathcal{P}(N)) \cup \{\heartsuit, L, R\}$$

The states of $\mathcal{B}$ are all the tuples $(\sigma, \gamma_1, H, \gamma_2, p)$ in which $\sigma \in \{\mathsf{L}, \mathsf{R}\}$ is the system, γ_1 is an N-configuration of the left system, H is a potted history and γ_2 is an N-configuration of the right system. The final component p tracks the state of the simulation using the following tokens and their meanings:

$\{\mathsf{A}\}$	(Attacker to play)
$\cup\, \{\mathsf{AL}\}$	(Attacker plays in L)
$\cup\, \{\mathsf{AR}\}$	(Attacker plays in R)
$\cup\, \{\mathsf{DL}[a] \mid a \in N\}$	(Defender plays a in L)
$\cup\, \{\mathsf{DR}[a] \mid a \in N\}$	(Defender plays a in R)
$\cup\, \{\mathsf{FL}[a] \mid a \in N\}$	(a forced in L)
$\cup\, \{\mathsf{FR}[a] \mid a \in N\}$	(a forced in R)
$\cup\, \{\mathsf{DL}[\tau] \mid \tau \in \delta \times N \times \mathcal{P}(N)\}$	(Defender plays τ in L)
$\cup\, \{\mathsf{DR}[\tau] \mid \tau \in \delta \times N \times \mathcal{P}(N)\}$	(Defender plays τ in R)

The following invariant is maintained throughout.

- At the start of simulating an $\mathcal{A}$-turn from position $((q_1, \rho_1, H), (q_2, \rho_2, H))$, the $\mathcal{B}$-game is in a configuration of shape:

  $$(\,(\mathsf{L}, \gamma_1, \phi \cdot H, \gamma_2, \mathsf{A}),\ (\mathsf{R}, \gamma_1, \phi \cdot H, \gamma_2, \mathsf{A})\,)$$

  where $\gamma_i = (q_i, \phi \cdot \rho_i)$ for some N-representation ϕ of $(\mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2), H)$.

In this way, the (N-representation of the) internal snapshot of the $\mathcal{A}$-game is synchronised between the two $\mathcal{B}$ systems and, according to the bookeeping, Attacker is to play. What makes the simulation complicated is that one cannot choose ϕ uniformly for the whole game. Instead, because of global freshness, a different representation must be sought for each new turn.

From the position described by the invariant, the $\mathcal{B}$-game simulates an Attacker move from the original game using two turns. The first turn allows Attacker to choose which of the two systems to play in. This is enabled by adding to $\mathcal{B}$, for all σ, γ_1, H, γ_2 all transitions of shapes:

$$(\sigma, \gamma_1, H, \gamma_2, \mathsf{A}) \xrightarrow{\mathsf{L}} (\sigma, \gamma_1, H, \gamma_2, \mathsf{AL})$$
$$(\sigma, \gamma_1, H, \gamma_2, \mathsf{A}) \xrightarrow{\mathsf{R}} (\sigma, \gamma_1, H, \gamma_2, \mathsf{AR})$$

The second turn simulates Attacker's choice of transition and is implemented by adding, for all σ, γ_1, H, γ_2, and all $T \in \delta$ and $a \in N$ the following transitions to $\mathcal{B}$:

$$(\sigma, \gamma_1, H, \gamma_2, \mathsf{AL}) \xrightarrow{(T,a)} (\sigma, \gamma_1', H, \gamma_2, \mathsf{DR}[a])$$
$$(\sigma, \gamma_1, H, \gamma_2, \mathsf{AR}) \xrightarrow{(T,a)} (\sigma, \gamma_1, H, \gamma_2', \mathsf{DL}[a])$$

subject to the constraint that, for the left case, $(\gamma_1, H) \xrightarrow{t,a} (\gamma_1', H \cup \{a\})$ is a transition of $\mathcal{A}$ induced by T and, for the right case, $(\gamma_2, H) \xrightarrow{t,a} (\gamma_2', H \cup \{a\})$ is a transition of $\mathcal{A}$ induced by T.

Next, the $\mathcal{B}$-game simulates a Defender move in the $\mathcal{A}$-game. This is accomplished over two turns, using Defender forcing. For the particular case in which Defender is to play on the right and has exactly two choices $\tau_1 = (T_1, a, H^1)$ and $\tau_2 = (T_2, a, H^2)$ to choose from, the Defender forcing circuit is shown in Figure 2. Each choice consists of a transition from δ, a letter from N and a potted history, subject to certain constraints. In this case, the game is forced into either the pair of configurations:

$$(\mathsf{L}, \gamma_1', H^1, \gamma_2^1, \mathsf{A})\ (\mathsf{R}, \gamma_1', H^1, \gamma_2^1, \mathsf{A})$$

where γ_2^1 is such that $(\gamma_2, H) \xrightarrow{t,a} (\gamma_2^1, H \cup \{a\})$ is a transition of $\mathcal{A}$ induced by T_1 and H^1 is a potting of $H \cup \{a\}$ wrt $\mathsf{rng}(\gamma_1') \cup \mathsf{rng}(\gamma_2^1)$. Or, analogously, the game is forced into:

$$(\mathsf{L}, \gamma_1', H^2, \gamma_2^2, \mathsf{A})\ (\mathsf{R}, \gamma_1', H^2, \gamma_2^2, \mathsf{A})$$

where γ_2^2 is such that $(\gamma_2, H) \xrightarrow{t,a} (\gamma_2^2, H \cup \{a\})$ is a transition of $\mathcal{A}$ induced by T_2 and H^2 is a potting of $H \cup \{a\}$ wrt $\mathsf{rng}(\gamma_1') \cup \mathsf{rng}(\gamma_2^2)$.

In general, there are not exactly two choices τ_1 and τ_2 from which Defender can choose from, so we generalise the gadget in the obvious way to support one choice

$$(\mathsf{R}, \gamma_1', H, \gamma_2, \mathsf{DR}[\tau_i]) \xrightarrow{\tau_i} (\mathsf{R}, \gamma_1', H^i, \gamma_2^i, \mathsf{A})$$

for each $\tau_i = (T_i, a, H^i)$ such that $(\gamma_2, H) \xrightarrow{t,a} (\gamma_2^i, H \cup \{a\})$ is a transition of $\mathcal{A}$ induced by T_i (a is fixed) and H^i is a potting of $H \cup \{a\}$ wrt $\mathsf{rng}(\gamma_1') \cup \mathsf{rng}(\gamma_2^i)$. Note that this generalisation includes the possibility that there are no choices for Defender, in which case the only transition in the gadget is $(\mathsf{L}, \gamma_1', H, \gamma_2, \mathsf{DR}[a]) \xrightarrow{\heartsuit} (\mathsf{L}, \gamma_1', H, \gamma_2, \mathsf{DR}[a])$. When Attacker has played on the right, so that Defender is to play on the left, we use a symmetric version of the gadget.

Given a binary relation $\approx$ over $\mathcal{B}$-configurations, let us write $\gamma_1 \approx_H \gamma_2$ whenever $((\mathsf{L}, \gamma_1, H, \gamma_2, \mathsf{A}), (\mathsf{R}, \gamma_1, H, \gamma_2, \mathsf{A})) \in \approx$. The central property of automaton construction is the following.

Lemma 52. *Let $\approx$ be a binary relation on $\mathcal{B}$-*

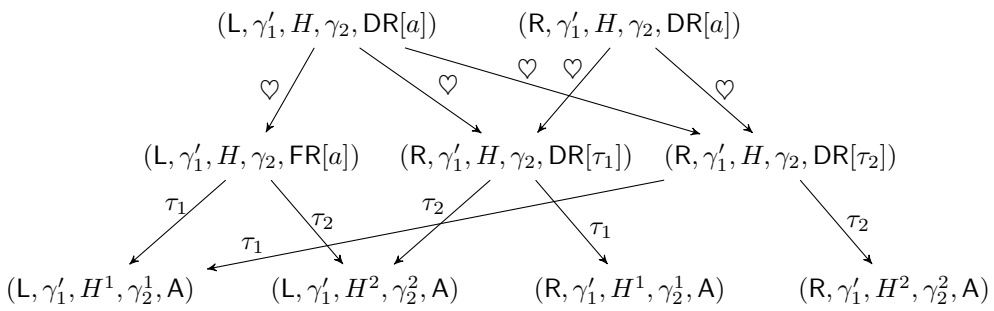


Fig. 2. Simulating Defender's move by forcing.

configurations whose elements are all of the form $(\mathsf{L}, \gamma_1, H, \gamma_2, \mathsf{A}), (\mathsf{R}, \gamma_1, H, \gamma_2, \mathsf{A})$. *Then there exists a* $\mathcal{S}(\mathcal{B})$*-bisimulation* $R \supseteq \approx$ *iff for all such elements of* R*:*

- *if* $(\gamma_1, H) \xrightarrow{t,a} (\gamma_1', H \cup \{a\})$ *then there is some* γ_2' *and some potting* $\hat{H}$ *of* $H \cup \{a\}$ *wrt* $\mathsf{rng}(\gamma_1') \cup \mathsf{rng}(\gamma_2')$ *such that:* $(\gamma_2, H) \xrightarrow{t,a} (\gamma_2', H \cup \{a\})$ *and* $\gamma_1' \approx_{\hat{H}} \gamma_2'$.
- *if* $(\gamma_2, H) \xrightarrow{t,a} (\gamma_2', H \cup \{a\})$ *then there is some* γ_1' *and some potting* $\hat{H}$ *of* $H \cup \{a\}$ *wrt* $\mathsf{rng}(\gamma_1') \cup \mathsf{rng}(\gamma_2')$ *such that:* $(\gamma_1, H) \xrightarrow{t,a} (\gamma_1', H \cup \{a\})$ *and* $\gamma_1' \approx_{\hat{H}} \gamma_2'$.

Proof: In the forward direction, assume that there exists a $\mathcal{S}(\mathcal{B})$-bisimulation R including $\approx$ and consider some $\gamma_1 \approx_H \gamma_2$. Let $(\gamma_1, H) \xrightarrow{t,a} (\gamma_1', H \cup \{a\})$ (*). By construction, if Defender is to win the Bisimulation game from this position, a necessary condition is that for every transition (i):

$$(\mathsf{L}, \gamma_1, H, \gamma_2, \mathsf{AL}) \xrightarrow{(T_1,a)} (\mathsf{L}, \gamma_1', H, \gamma_2, \mathsf{DR}[a])$$

there is a corresponding transition (ii):

$$(\mathsf{R}, \gamma_1', H, \gamma_2, \mathsf{DR}[(T_2, a, H^2)]) \xrightarrow{(T_2,a,H^2)} (\mathsf{R}, \gamma_1', H^2, \gamma_2', \mathsf{A})$$

It follows from $(*)$ that there is a transition (i) and therefore also a transition (ii). Necessarily then $(\gamma_2, H) \xrightarrow{t,a} (\gamma_2', H \cup \{a\})$ and it follows from the construction that there will be such a transition for each potted history satisfying the constraints (of which there is always at least one). The other case is symmetric.

In the backward direction, assume the conditions are satisfied and we show that such an R exists by arguing that every pair of configurations in $\approx$ is a winning position for Defender in the induced bisimulation game. To see this simply observe that, by construction, the only way that Attacker can win in the bisimulation game is for the game to reach a position of shape $((\mathsf{L}, \gamma_1', H, \gamma_2, \mathsf{DR}[a]), (\mathsf{R}, \gamma_1', H, \gamma_2, \mathsf{DR}[a]))$ (or symmetrically with final component $\mathsf{DL}[a]$) and for Defender to have zero choices in the Defender forcing gadget. However, this can only happen if Attacker was able to choose some transition

$$(\mathsf{L}, \gamma_1, H, \gamma_2, \mathsf{AL}) \xrightarrow{(T_1,a)} (\mathsf{L}, \gamma_1', H, \gamma_2, \mathsf{DR}[a])$$

and there is no corresponding transition for Defender to force. However, it follows from the assumption that there is always some choice of corresponding transition. Note that if a transition $(\gamma_2, H) \xrightarrow{t,a} (\gamma_2', H \cup \{a\})$ is valid then there is always some choice of $\hat{H}$ satisfying the constraints. The other case is symmetrical. ∎

From a $\mathcal{S}(\mathcal{B})$-bisimulation and an appropriate N-representation one may construct a $\mathcal{S}(\mathcal{A})$-bisimulation.

Lemma 53. *Suppose* $\approx$ *is a* $\mathcal{S}(\mathcal{B})$*-bisimulation. Then let* R' *contain* $((q_1, \rho_1, H), (q_2, \rho_2, H))$ *just if:*

(i) $(q_1, \phi \cdot \rho_1) \approx_{\phi \cdot H} (q_2, \phi \cdot \rho_2)$

(ii) and ϕ *is any* N*-representation of* $(\mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2), H)$.

Then R' *is a* $\mathcal{S}(\mathcal{A})$*-bisimulation.*

Proof: Suppose $((q_1, \rho_1, H), (q_2, \rho_2, H)) \in R'$ and $(q_1, \rho_1, H) \xrightarrow{t,a} (q_1', \rho_1'[i \mapsto a], H \cup \{a\})$ by some transition $T_1 \in \delta$. It follows from Lemma 49 that therefore $(q_1, \phi \cdot \rho_1, \phi \cdot H) \xrightarrow{t,\phi(a)} (q_1', (\phi \cdot \rho_1')[i \mapsto \phi(a)], (\phi \cdot H) \cup \{\phi(a)\})$. Since $\approx$ is a bisimulation, it follows from Lemma 52 that there is some $(q_2, \phi \cdot \rho_2, \phi \cdot H) \xrightarrow{t,\phi(a)} (q_2', (\phi \cdot \rho_2')[j \mapsto \phi(a)], (\phi \cdot H) \cup \{\phi(a)\})$, and some $\hat{H}$ a potting of $(\phi \cdot H) \cup \{\phi(a)\}$ wrt $\mathsf{rng}(\phi \cdot \rho_1') \cup \mathsf{rng}(\phi \cdot \rho_2') \cup \{a\}$ such that $(q_1', (\phi \cdot \rho_1')[i \mapsto \phi(a)]) \approx_{\hat{H}} (q_2', (\phi \cdot \rho_2')[j \mapsto \phi(a)])$. It follows again from Lemma 49 that therefore there is some $(q_2, \rho_2, H) \xrightarrow{t,a} (q_2', \rho_2'[j \mapsto a], H \cup \{a\})$. Next observe that, by construction, $N \setminus \hat{H} \neq \emptyset$ and, if $H \setminus \mathsf{rng}(\rho_1') \cup \mathsf{rng}(\rho_2') \neq \emptyset$, then $\hat{H} \setminus (\mathsf{rng}(\phi \cdot \rho_1') \cup \mathsf{rng}(\phi \cdot \rho_2') \cup \{\phi(a)\}) \neq \emptyset$. Consequently, it is possible to find surjections:

$$\begin{array}{lcl} f_1 & : & (\mathcal{D} \setminus (H \cup \{a\})) \to (N \setminus \hat{H}) \\ f_2 & : & (H \cup \{a\} \setminus \mathsf{rng}(\rho_1') \cup \mathsf{rng}(\rho_2') \cup \{a\}) \\ & & \quad \to (\hat{H} \setminus \mathsf{rng}(\phi \cdot \rho_1') \cup \mathsf{rng}(\phi \cdot \rho_2') \cup \{\phi(a)\} \end{array}$$

and construct the particular bijection:

$$f_3 : \mathsf{rng}(\rho_1') \cup \mathsf{rng}(\rho_2') \cup \{a\} \to \mathsf{rng}(\phi \cdot \rho_1') \cup \mathsf{rng}(\phi \cdot \rho_2') \cup \{\phi(a)\}$$

given by $f_3 = \phi \restriction \mathsf{rng}(\rho_1') \cup \mathsf{rng}(\rho_2') \cup \{a\}$. It follows that $\phi = f_1 \cup f_2 \cup f_3$ is an N-representation of $(\mathsf{rng}(\rho_1') \cup \mathsf{rng}(\rho_2') \cup \{a\}, H)$ such that $\phi' \cdot (\rho_1'[i \mapsto a]) = (\phi \cdot \rho_1')[i \mapsto \phi(a)]$, $\phi' \cdot (\rho_2'[i \mapsto a]) = (\phi \cdot \rho_2')[j \mapsto \phi(a)]$ and $\phi' \cdot H = \hat{H}$ as required. The case in which the second component makes a transition is symmetrical. ∎

Similarly, from a $\mathcal{S}(\mathcal{A})$-bisimulation and an appropriate N-representation, one may construct a $\mathcal{S}(\mathcal{B})$-bisimulation.

Lemma 54. *Suppose R is a $\mathcal{S}(\mathcal{A})$-bisimulation. Let $\approx$ relate $(\mathsf{L}, q_1, \phi{\cdot}\rho_1, \phi{\cdot}H, q_2, \phi{\cdot}\rho_2, \mathsf{A})$ to $(\mathsf{R}, q_1, \phi{\cdot}\rho_1, \phi{\cdot}H, q_2, \phi{\cdot}\rho_2, \mathsf{A})$ just if:*

(i) $((q_1, \rho_1, H), (q_2, \rho_2, H)) \in R$

(ii) and ϕ is an N-representation of $(\mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2), H)$.

Then there exists a $\mathcal{B}$-simulation containing $\approx$.

Proof: By Lemmma 52 suppose $(q_1, \phi \cdot \rho_1, \phi \cdot H) \xrightarrow{t,a} (q_1', (\phi \cdot \rho_1')[i \mapsto a], (\phi \cdot H) \cup \{a\})$. It follows from Lemma 52 and surjectivity of representations that there is a corresponding $(q_1, \rho_1, H) \xrightarrow{t,b} (q_1', \rho_1'[i \to b], H \cup \{b\})$. Since R is a bisimulation, it follows that there is some $(q_2, \rho_2, H) \xrightarrow{t,b} (q_2', \rho_2'[j \to b], H \cup \{b\})$ and $(q_1', \rho_1'[i \to b], H)$ is related to $(q_2', \rho_2'[j \to b], H \cup \{b\})$ by R. It follows from Lemma 49 that there is a corresponding $(q_2, \phi{\cdot}\rho_2, \phi{\cdot}H) \xrightarrow{t,a} (q_2', (\phi{\cdot}\rho_2')[j \to a], (\phi \cdot H) \cup \{a\})$. Let $\hat{H}$ be any potting of $(\phi \cdot H) \cup \{a\}$ wrt $\mathsf{rng}(\phi \cdot \rho_1') \cup \mathsf{rng}(\phi \cdot \rho_2') \cup \{a\}$. Then it follows that $N \setminus \hat{H} \neq \emptyset$ and $(H \cup \{b\}) \setminus (\mathsf{rng}(\rho_1') \cup \mathsf{rng}(\rho_2') \cup \{b\}) \neq \emptyset$ implies $\hat{H} \setminus (\mathsf{rng}(\phi{\cdot}\rho_1') \cup \mathsf{rng}(\phi{\cdot}\rho_2') \cup \{a\}) \neq \emptyset$. Consequently it is possible to find surjections:

$$\begin{array}{rcl} f_1 & : & (\mathcal{D} \setminus (H \cup \{b\})) \to (N \setminus \hat{H}) \\ f_2 & : & (H \cup \{b\} \setminus \mathsf{rng}(\rho_1') \cup \mathsf{rng}(\rho_2') \cup \{b\}) \\ & & \quad \to (\hat{H} \setminus \mathsf{rng}(\phi \cdot \rho_1') \cup \mathsf{rng}(\phi \cdot \rho_2') \cup \{a\} \end{array}$$

and construct the particular bijection:

$$f_3 : \mathsf{rng}(\rho_1') \cup \mathsf{rng}(\rho_2') \cup \{b\} \to \mathsf{rng}(\phi \cdot \rho_1') \cup \mathsf{rng}(\phi \cdot \rho_2') \cup \{a\}$$

given by $f_3 = \phi \restriction \mathsf{rng}(\rho_1') \cup \mathsf{rng}(\rho_2') \cup \{b\}$. It follows that $\phi = f_1 \cup f_2 \cup f_3$ is an N-representation of $(\mathsf{rng}(\rho_1') \cup \mathsf{rng}(\rho_2') \cup \{b\}, H)$ such that $\phi' \cdot (\rho_1'[i \mapsto b]) = (\phi \cdot \rho_1')[i \mapsto a]$, $\phi' \cdot (\rho_2'[i \mapsto b]) = (\phi \cdot \rho_2')[j \mapsto a]$ and $\phi' \cdot H = \hat{H}$ as required. The case in which the second component makes a transition is symmetrical. Hence, we conclude by Lemma 52. ∎

Consequently, we can prove the main result:

Theorem 55. *Suppose ϕ is an N-representation of $(\mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2), H)$. Then $(q_1, \rho_1, H) \sim (q_2, \rho_2, H)$ iff $(q_1, \phi \cdot \rho_1) \sim_{\phi \cdot H} (q_2, \phi \cdot \rho_2)$.*

Proof: Suppose $(q_1, \rho_1, H) \sim (q_2, \rho_2, H)$ in $\mathcal{S}(\mathcal{A})$. Then it follows from Lemma 54 that there is some $\mathcal{S}(\mathcal{B})$-bisimulation $\approx$ such that $(q_1, \phi \cdot \rho_1) \approx_{\phi \cdot H} (q_2, \phi \cdot \rho_2)$. Conversely, suppose that $(q_1, \phi \cdot \rho_1) \sim_{\phi \cdot H} (q_2, \phi \cdot \rho_2)$ in $\mathcal{S}(\mathcal{B})$. It follows from Lemma 53 that there is some $\mathcal{S}(\mathcal{B})$-bisimulation R such that $((q_1, \rho_1, H), (q_2, \rho_2, H)) \in R$. ∎

B. Proof of Proposition 10

Definition 56. An ***alternating linear bounded automaton*** (ALBA) is a tuple $\mathcal{A} = \langle \Gamma, Q_\forall, Q_\exists, q_0, q_{\text{acc}}, q_{\text{rej}}, \delta \rangle$. We write $Q = Q_\forall \uplus Q_\exists$. The components are:

- A finite tape alphabet Γ containing end-of-tape markers $\triangleleft$ and $\triangleright$.
- A finite set of universal states $Q_\forall$.
- A disjoint, finite set of existential states $Q_\exists$.
- Distinguished initial state $q_0 \in Q$.
- Disjoint accepting and rejecting states $q_{\text{acc}}, q_{\text{rej}} \notin Q$.
- A transition function $\delta : Q \times \Gamma \to (Q \times \Gamma \times \{-1, +1\})^*$, satisfying the following properties: (i) if $(q', a, d) \in \mathsf{rng}(\delta(q, \triangleright))$ then $a = \triangleright$ and $d = +1$; (ii) if $(q', a, d) \in \mathsf{rng}(\delta(q, \triangleleft))$ then $a = \triangleleft$ and $d = -1$; (iii) if $(q', a, d) \in \delta(q, b)$ then $b \in \Gamma \setminus \{\triangleleft, \triangleright\}$ implies $a \in \Gamma \setminus \{\triangleleft, \triangleright\}$.

A ***configuration*** of such a machine is a triple $c = (q, k, t)$ with q a state, t the current tape contents and k the index of the cell currently under the head of the machine. We assume that the tape contents is of the form $\triangleright a_1 \cdots a_n \triangleleft$ for some letters $a_i \in \Gamma \setminus \{\triangleleft, \triangleright\}$. We write $t(k)$ for the contents of cell k of tape t. We say that a configuration (q, k, t) is ***accepting*** (respectively ***rejecting***, ***universal***, ***existential***) just if $q = q_{\text{acc}}$ (respectively $q = q_{\text{rej}}$, $q \in Q_\forall$, $q \in Q_\exists$).

A configuration (q_1, k_1, t_1) can make a transition to a ***successor*** (q_2, k_2, t_2) just if there is $a \in \Gamma \setminus \{\triangleleft, \triangleright\}$ and $d \in \{-1, +1\}$ such that $(q_2, a, d) \in \mathsf{rng}(\delta(q_1, t_1(k_1)))$ and $k_2 = k_1 + d$ and $t_2 = t_1[k_1 \mapsto a]$.

Given an input $w \in \Gamma \setminus \{\triangleleft, \triangleright\}$, a ***computation tree on*** w for such a machine is an unordered tree labelled by configurations which additionally satisfies the following conditions:

- The tree is rooted at $(q_0, 1, \triangleright w \triangleleft)$.
- If a universal configuration c labels some node of the tree then this node has one child for each possible successor to c.
- If an existential configuration c labels some node of the tree then this node has exactly one child which can be any successor to c.

We say that an input w is ***accepted*** just if every leaf of the computation tree on w is accepting.

The problem of deciding whether a given input is accepted by a given ALBA is well known to be EXPTIME-complete.

Definition 57. The problem ALBA-MEM is, given an ALBA $\mathcal{M}$ and an input w, to determine whether w is accepted by $\mathcal{M}$.

For the purposes of the argument, we will assume without loss of generality that $\Gamma \setminus \{\triangleleft, \triangleright\} = \{0, 1\}$ and, for all (q, a), $|\delta(q, a)| = 2$. Thus all choices presented by the alternation are binary — a unary choice may be represented by having $\delta(q, a)(1) = \delta(q, a)(2)$. Starting from an instance of the ALBA-MEM problem $\langle \mathcal{M}, w \rangle$, we construct a bisimulation problem for RA($S\#$) in which two configurations are bisimilar iff $\mathcal{M}$ accepts w.

Given an instance $\langle \mathcal{M}, w \rangle$ of the ALBA-MEM problem, we construct a $2|w|+2$ register RA($S\#$) $\mathcal{A}^w_{\mathcal{M}}$ whose induced bisimulation game simulates the computations of $\mathcal{M}$ by having Attacker choose successor configurations from universal states and Defender choose successor configurations from existential states. A configuration of a computation of $\mathcal{M}$ will be represented, in duplicate, by a pair of configurations of $\mathcal{A}^w_{\mathcal{M}}$ which together make up a single configuration of the bisimulation game. These configurations will track the current state of $\mathcal{M}$ and the current position of the head of $\mathcal{M}$ in their state and

the current tape contents of $\mathcal{M}$ will be represented by their current register assignment $\mathcal{A}^w_{\mathcal{M}}$.

Tape encoding. In particular, the last $2|w|$ registers of $\mathcal{A}^w_{\mathcal{M}}$ will be used to encode the non-endmarker tape contents of $\mathcal{M}$. For this purpose we use the following encoding: tape cell $k \in [2, |w|+1]$ contains 0 iff register $2k-1$ is empty and register $2k$ contains a letter; similarly tape cell $k \in [2, |w|+1]$ contains 1 iff register $2k-1$ contains a letter and register $2k$ is empty. The first two registers 1 and 2 are used to help implement a simulation of alternation and will never be empty.

We build the states of this automaton from the transition relation of $\mathcal{M}$, tape cell indices and special tags L, R, 0, 1 and 2. The states are as follows:

- For each $q \in Q$, $k \in [1, |w|+2]$, $a \in \Gamma$, $C \in \{L, R\}$: states (q, k, C), (q, k, a, C).
- For each $q \in Q_\forall$, $k \in [1, |w|+2]$: states (q, k, L) and (q, k, R)
- For each $q \in Q_\exists$, $k \in [1, |w|+2]$: states (q, k, L), (q, k, R), $(q, k, 0)$, $(q, k, 1)$ and $(q, k, 2)$.
- For each $(q, a, d) \in \bigcup \mathsf{rng}(\delta)$, $k \in [1, |w|+2]$: states (k, q, a, d, L) and (k, q, a, d, R).
- For each $k \in [1, |w|+2]$, $C \in \{L, R\}$: states (q_{acc}, k, C) and (q_{rej}, k, C).

Let us write a typical state of $\mathcal{A}^w_{\mathcal{M}}$ as p. Given a state p, we write $p[t]$ for the tuple p with its final component replaced by tag $t \in \{L, R, 0, 1, 2\}$.

Let us take any $2|w|+2$-register assignment ρ_I that encodes the initial input w as above and also assigns some letters to registers 1 and 2. We define the construction of $\mathcal{A}^w_{\mathcal{M}}$ to ensure that configurations $((q_0, 1, L), \rho_I)$ and $((q_0, 1, R), \rho_I)$ are bisimilar iff $\mathcal{M}$ accepts w.

We motivate the construction via the bisimulation game that it induces, a configuration of which is a pair of configurations $((p_1, \rho_1), (p_2, \rho_2))$ of $\mathcal{A}^w_{\mathcal{M}}$. We define $\mathcal{A}^w_{\mathcal{M}}$ so that the following invariant holds for configurations in the induced bisimulation game: every configuration in the game has shape $((p_1, \rho), (p_2, \rho))$ in which p_1 and p_2 are either identical or identical except for the final tag component. The idea is that when the two configurations are of the form $((q, k, C), \rho)$, with $C \in \{L, R\}$, the play is simulating a configuration of $\mathcal{M}$ which is in state q, with the head over tape cell k and the tape contents itself encoded by the first $2|w|$ registers of ρ. When the two configurations are of the form (q, k, a, C) then furthermore the simulation as decoded symbol a as being the current contents of the tape at cell k. When the two configurations are of the form $((k, q, a, d, C), \rho)$, then the play is part of the way through simulating a transition, as given by $(q, a, d) \in \bigcup \mathsf{rng}(\delta)$. Finally, when the configurations are of the form $((k, q, i)\, \rho)$, for $i \in \{0, 1, 2\}$, the play is part of the way through simulating a transition which is, in particular, a transition from an existential state and for which defender forcing is required.

We will not require the use of any tags (*cf.* data words) in our construction, so we assume that Σ is a unary alphabet and omit this component in transitions. In order to describe

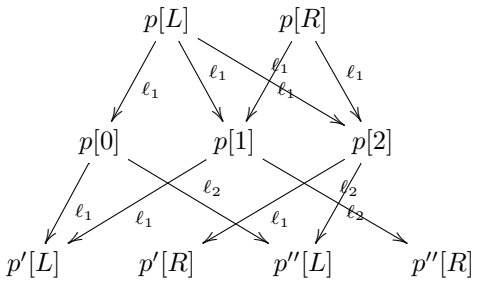


Fig. 3. $\mathsf{DF}(p)(p', p'')$

the transition relation of the automaton we will make use of a gadget to implement defender forcing. Since the configurations of the induced bisimulation game are guaranteed, by the invariant, to have the same register contents, we are able to instantiate the general construction of [10], in which Attacker is punished for making choices inconsistent with Defender's wishes by allowing Defender to move his system into a configuration identical with that of Attacker. The gadget $\mathsf{DF}(p)(p', p'')$ ensures that, when the game configuration consists of two system configurations of shape $(p[L], \rho)$ and $(p[R], \rho)$, then Defender can force the play so that the game enters a configuration consisting of either the two system configurations $(p'[L], \rho)$ and $(p'[R], \rho)$, or the two system configurations $(p''[L], \rho)$ and $(p''[R], \rho)$. The gadget is shown in Figure 3, in which the labels on the transitions are given by $\ell_1 = \{1\}, 0, \emptyset$ and $\ell_2 = \{2\}, 0, \emptyset$. It is by this defender forcing gadget that we will be able to ensure that the two players correctly simulate existential choices made by $\mathcal{M}$, essentially by allowing Defender to make the choice.

We describe the transitions of $\mathcal{A}^w_{\mathcal{M}}$ as part of a general description of how the induced bisimulation game simulates $\mathcal{M}$. Recall that a configuration of the form $((q, k, C), \rho)$ is used to simulate $\mathcal{M}$ in operating in state q with the head over cell k of tape encoded by ρ. Simulating a transition of $\mathcal{M}$ from this configuration is broken into three parts. In the first part the tape encoding in ρ is decoded to obtain the letter a under the head. In the second part a successor is chosen from among $\delta(q, a)$. In the third part the game over $\mathcal{A}^w_{\mathcal{M}}$ moves into a configuration representing the simulation of the successor in $\mathcal{M}$.

Decoding the tape. The decoding of the tape is split into two cases, depending on whether the head of the machine being simulated is over an endmarker or not. If $k \in \{1, |w|+1\}$ then the head is over an endmarker, and the contents of cell k is completely determined by k. Hence, in such cases we use transitions of shape:

$$(q, k, C) \xrightarrow{\{1\},0,\emptyset} (q, k, a, C)$$

where $k = 1$ implies $a = \triangleright$ and $k = |w|+1$ implies $a = \triangleleft$. Otherwise $k \in [2, |w|]$ and the head is over a cell which is encoded in the way described above. To decode it we use

pairs of transitions of shape:

$$(q,k,C) \xrightarrow{\{2k\},0,\{2k\}} (q,k,0,C)$$
$$(q,k,C) \xrightarrow{\{2k-1\},0,\{2k-1\}} (q,k,1,C)$$

since the tape encoding has cell k represented by cells $2k-1$ and $2k$, of which exactly one contains a letter. Hence, from a configuration $((q,k,C),\rho)$ with ρ an encoding of the tape, exactly one of these two transitions will be applicable. The state (q,k,i,C) records the value i that was read, and the tape contents after the transition empties the register that stored the letter, so that both cells $k-1$ and k are empty. This prepares the way for writing a new value in the third part.

Choosing a transition from a universal state. If $q \in Q_\forall$ then it is up to Attacker to choose the successor. Hence, we use the following transitions:

$$\{(q,k,a,C) \xrightarrow{\{l\},0,\emptyset} (k,q',b,d,C) \mid \delta(q,a)(l) = (q',b,d)\}$$

Since, in each round of the bisimulation game, Attacker gets to choose the next transition of $\mathcal{A}^w_\mathcal{M}$ first, he will choose which transition of $\mathcal{M}$ is simulated. Since his choice is communicated by announcing (in the $\mathcal{A}^w_\mathcal{M}$-transition label) the index $l \in \{1,2\}$ of the $\mathcal{M}$-transition, Defender will be forced to choose the same transition in the complementary system.

Choosing a transition from an existential state. If $q \in Q_\exists$ then it is up to Defender to choose the successor. In this case we use an instance of the defender forcing gadget:

$$\mathsf{DF}(q,k,i,\_)((k,q_1,b_1,d_1,\_),(k,q_2,b_2,d_2,\_))$$

where $\delta(q,i) = (q_1,b_1,d_1)\,(q_2,b_2,d_2)$. Note that the tags on the states here are irrelevant since they will be replaced by the construction of the gadget. This ensures that Defender can steer the simulation into her choice whilst maintaining the invariant about the shape of configurations.

Computing the successor. Moving the configuration of $\mathcal{A}^w_\mathcal{M}$ into a state that simulates the successor according to the chosen transition of $\mathcal{M}$ involves changing the contents of the current register assignment to reflect the action of $\mathcal{M}$ writing to the tape. Hence, we split the construction based on whether or not the cell k contains an endmarker. If $k \in \{1, |w|+1\}$ then the cell under the head contains an endmarker and hence the register assignment should not change, but only the state since that contains the information about the current position of the head. To this end we use transitions of the following shape:

$$(k,q',b,d,C) \xrightarrow{\{1\},0,\emptyset} (q',k+d,C)$$

Otherwise $k \in [2,|w|]$ and the registers should be updated according to the encoding described above. We use the transitions of shape:

$$(k,q',b,d,C) \xrightarrow{\emptyset,2k-b,\emptyset} (q',k+d,C)$$

Accepting and rejecting states. If the simulation reaches an accepting state then Defender should win. We organise for this to happen by forbidding any transition out of any state of shape (q_{acc},k,C). In this way, any two configurations that are both in states of this form are trivially bisimilar since neither can perform an action. Conversely Attacker should win if the simulation reaches a rejecting state. We organise for this to happen by transitions of the following shape:

$$(q_{\text{rej}},k,L) \xrightarrow{1,0,\emptyset} (q_{\text{rej}},k,L)$$

Notice that such transitions only occur in those states that are tagged L. By construction, when the simulation arrives at a rejecting state, one configuration will in such a state tagged with L and the other with R and it follows that the two configurations will not be bisimilar.

Lemma 58. *Given an ALBA $\mathcal{M}$ and input w, $\mathcal{M}$ accepts w iff Defender has a winning strategy in the bisimulation game for $\langle \mathcal{S}_{\mathcal{A}^w_\mathcal{M}}, ((q_0,1,L),\rho_I)((q_0,1,R),\rho_I)\rangle$, where ρ_I is a register assignment encoding w in the way described above.*

Proof: First notice that, by construction, there are only two ways that Defender can win a play of the bisimulation game:

(i) By Attacker choosing a move in the defender forcing gadget which results in a punishment response from Defender so that every game configuration that follows in the play is of shape $((p,\rho),(p,\rho))$ whose components are trivially bisimilar.
(ii) By the play reaching a game configuration in which the two component configurations are of shape $((q,k,L),\rho)$ and $((q,k,R),\rho)$ for $q = q_{\text{acc}}$, which are bisimilar by construction.

In the forward direction, assume that $\mathcal{M}$ accepts w. Then there is a computation tree T for w in which every leaf is accepting. Hence Defender can win every play of the corresponding bisimulation game by using T as a representation of a winning strategy. In particular, for any given play there are two possibilities. If Attacker plays badly inside a defender forcing gadget and is punished then the result is (i) above. Otherwise, as long as Defender makes choices consistent with T then every play will eventually reach a configuration which simulates $\mathcal{M}$ in accepting state q_{acc}. By construction, the corresponding game configuration must have component configurations of shape $((q_{\text{acc}},k,L),\rho)$ and $((q_{\text{acc}},k,R),\rho)$ and Defender wins as described in (ii).

In the backward direction, assume that Defender has a winning strategy for the bisimulation game. Then, since this strategy must specify which transition to choose when simulating a computation from an existential state, the strategy can be used to build a computation tree T for $\mathcal{M}$ on w. Since, by construction, Attacker can always avoid being punished whilst playing in a defender forcing gadget, it follows that W must allow Defender to win any such play by the criterion (ii). Hence, every simulation which follows W ends in an accepting state and it follows that every leaf of T is accepting. ■

APPENDIX C
PROOFS FROM SECTION IV

A. Proof of Lemma 13

The second part of the result is the content of the following two lemmata.

Lemma 59. *For all $i \in \omega$, $\overset{i+1}{\sim} \subseteq \overset{i}{\sim}$*

Proof: The proof is by induction on i. When i is 0, the result is trivial as $\overset{i}{\sim}$ is the universe. When $i = k+1$, assume $(q_1, S_1) \overset{k+2}{\sim}_\tau (q_2, S_2)$. It follows by definition that $(q_1, S_1, \tau, q_2, S_2)$ and $(q_2, S_2, \tau^{-1}, q_1, S_1)$ satisfy the (SYS) conditions in $\overset{k+1}{\sim}$. It follows from the induction hypothesis that $\overset{k+1}{\sim} \subseteq \overset{k}{\sim}$ since set union is monotonic. Hence, they also satisfy the (SYS) conditions in $\overset{k}{\sim}$ whence $(q_1, S_1) \overset{k+1}{\sim}_\tau (q_2, S_2)$. ∎

Lemma 60. $\bigcap_{i\in\omega} \overset{i}{\sim} = \overset{\mathsf{s}}{\sim}$

Proof: We start with the $\supseteq$ direction and argue that, for all $i \in \omega$, $\overset{\mathsf{s}}{\sim}$ is a lower bound on $\overset{i}{\sim}$. The proof is by induction on i. When $i = 0$ the result is trivial. When $i = k+1$, assume $(q_1, S_1) \overset{\mathsf{s}}{\sim}_\tau (q_2, S_2)$. We wish to show that it and its inverse satisfy the (SYS) conditions in $\overset{k}{\sim}$. By definition, they satisfy the (SYS) conditions in $\overset{\mathsf{s}}{\sim}$. Now observe that it follows from the induction hypothesis that $\overset{\mathsf{s}}{\sim} \subseteq \overset{k}{\sim}$, so the tuples satisfy the (SYS) conditions in $\overset{k}{\sim}$.

For the $\subseteq$ direction, we argue that the left-hand side is a symbolic bisimulation. To see this, assume $(q_1, S_1, \tau, q_2, S_2) \in \bigcap_{i\in\omega} \overset{i}{\sim}$ so that $(q_1, S_1, \tau, q_2, S_2)$ and its inverse satisfy the (SYS) conditions in $\overset{i}{\sim}$, for all $i \in \omega$. However, this is just to say that they satisfies the (SYS) conditions in $\bigcap_{i\in\omega} \overset{i}{\sim}$. ∎

For the first part, we show a correspondence between bisimulations and symbolic bisimulations from which the result follows.

bisim → s-bisim. Let R be a bisimulation on $\mathcal{A}$. We claim that the relation $R' \subseteq \mathcal{U}$,

$$R' = \{\, (q_1, S_1, \sigma, q_2, S_2) \mid \exists \rho_1, \rho_2.\ (q_1, \rho_1) R (q_2, \rho_2) \\ \wedge\, \sigma = \rho_1; \rho_2^{-1} \wedge \mathsf{dom}(\rho_i) = S_i \,\}$$

is a symbolic bisimulation. For the latter (by symmetry in the definition) it suffices to show that R' is a symbolic simulation. So suppose that $(q_1, S_1, \sigma, q_2, S_2) \in R'$ due to some $(q_1, \rho_1) R (q_2, \rho_2)$. Let $q_1 \xrightarrow{t,i} q_1'$ for some $i \in S_1$. Then, $(q_1, \rho_1) \xrightarrow{t,a} (q_1', \rho_1)$ with $a = \rho_1(i)$ and, hence, $(q_2, \rho_2) \xrightarrow{t,a} (q_2', \rho_2')$ with $(q_1', \rho_1) R (q_2', \rho_2')$.

- If $i \in \mathsf{dom}(\sigma)$ then $a = \rho_2(\sigma(i))$ and therefore the above transition is due to some $q_2 \xrightarrow{t,\sigma(i)} q_2'$, and $\rho_2' = \rho_2$. Hence, $(q_1', S_1) R'_\sigma (q_2', S_2)$.
- If $i \notin \mathsf{dom}(\sigma)$ then the transition is due to some $q_2 \xrightarrow{t,j^\bullet} q_2'$, and $\rho_2' = \rho_2[j \mapsto a]$. Hence, since $\sigma[i \mapsto j] = \rho_1; (\rho_2[j \mapsto a])^{-1}$ and $\mathsf{dom}(\rho_2') = S_2[j]$, we have $(q_1', S_1) R'_{\sigma[i\mapsto j]} (q_2', S_2[j])$.

Now let $q_1 \xrightarrow{t,i^\bullet} q_1'$. For each $a \notin \mathsf{rng}(\rho_1)$, $(q_1, \rho_1) \xrightarrow{t,a} (q_1', \rho_1')$ with $\rho_1' = \rho_1[i \mapsto a]$ and, hence, there is some $(q_2, \rho_2) \xrightarrow{t,a} (q_2', \rho_2')$ with $(q_1', \rho_1') R (q_2', \rho_2')$.

- Select some $a \notin \mathsf{rng}(\rho_2)$. Then, the transition above is due to some $q_2 \xrightarrow{t,j^\bullet} q_2'$, and $\rho_2' = \rho_2[j \mapsto a]$. Moreover, since $\sigma[i \mapsto j] = \rho_1[i \mapsto a]; (\rho_2[j \mapsto a])^{-1}$, $\mathsf{dom}(\rho_1') = S_1[i]$ and $\mathsf{dom}(\rho_2') = S_2[j]$, we have $(q_1', S_1[i]) R'_{\sigma[i\mapsto j]} (q_2', S_2[j])$.
- Let $j \in S_2 \setminus \mathsf{rng}(\sigma)$. Then, we can take a to be $\rho_2(j)$, so the transition is due to some $q_2 \xrightarrow{t,j} q_2'$, and $\rho_2' = \rho_2$. We moreover have $(q_1', S_1[i]) R'_{\sigma[i\mapsto j]} (q_2', S_2)$.

s-bisim → bisim. Let R be a symbolic bisimulation on $\mathcal{A}$. We claim that the relation

$$R' = \{\, ((q_1, \rho_1), (q_2, \rho_2)) \mid (q_1, S_1) R_\sigma (q_2, S_2) \\ \wedge\, \sigma = \rho_1; \rho_2^{-1} \wedge S_i = \mathsf{dom}(\rho_i) \,\}$$

is a bisimulation, for which it suffices to show that R' is a simulation. So suppose that $((q_1, \rho_1), (q_2, \rho_2)) \in R'$ due to some $(q_1, S_1) R_\sigma (q_2, S_2)$, and let $(q_1, \rho_1) \xrightarrow{t,a} (q_1', \rho_1')$ for some $(t, a) \in \Sigma \times \mathcal{D}$. If $a \in \mathsf{rng}(\rho_1)$, say $a = \rho_1(i)$, then $q_1 \xrightarrow{t,i} q_1'$ and $\rho_1' = \rho_1$. We distinguish two cases:

- If $a \in \mathsf{rng}(\rho_2)$ then $i \in \mathsf{dom}(\sigma)$, so $q_2 \xrightarrow{t,\sigma(i)} q_2'$ and $(q_1', S_1) R_\sigma (q_2', S_2)$. Hence, $(q_2, \rho_2) \xrightarrow{t,a} (q_2', \rho_2)$ and $(q_1', \rho_1) R' (q_2', \rho_2)$.
- If $a \notin \mathsf{rng}(\rho_2)$ then $i \in S_1 \setminus \mathsf{dom}(\sigma)$, so $q_2 \xrightarrow{t,j^\bullet} q_2'$ and $(q_1', S_1) R_{\sigma[i\mapsto j]} (q_2', S_2[j])$. Hence, $(q_2, \rho_2) \xrightarrow{t,a} (q_2', \rho_2[j \mapsto a])$ and $(q_1', \rho_1) R' (q_2', \rho_2[j \mapsto a])$.

If $a \notin \mathsf{rng}(\rho_1)$ then there is $q_1 \xrightarrow{t,i^\bullet} q_1'$ such that $\rho_1' = \rho_1[i \mapsto a]$.

- If $a \notin \mathsf{rng}(\rho_2)$ then, since $q_2 \xrightarrow{t,j^\bullet} q_2'$ with $(q_1', S_1[i]) R_{\sigma[i\mapsto j]} (q_2', S_2[j])$, we obtain $(q_2, \rho_2) \xrightarrow{t,a} (q_2', \rho_2[j \mapsto a])$ and $(q_1', \rho_1') R' (q_2', \rho_2[j \mapsto a])$.
- If $a \in \mathsf{rng}(\rho_2)$, say $a = \rho_2(j)$, then $j \in S_2 \setminus \mathsf{rng}(\sigma)$. Hence, $q_2 \xrightarrow{t,j} q_2'$ with $(q_1', S_1[i]) R_{\sigma[i\mapsto j]} (q_2', S_2)$, from which we get $(q_2, \rho_2) \xrightarrow{t,a} (q_2', \rho_2)$ and $(q_1', \rho_1') R' (q_2', \rho_2)$.

□

B. Proof of Lemma 14

We first observe that, since $R = R^{-1}$, $Cl(R) = Cl^-(R)$ where, for any relation X, we let $Cl^-(X)$ be the smallest relation that contains X and is closed under the rules (ID), (TR) and (EXT) above. Let us set $R' = Cl^-(R)$ and $P' = Cl(P)$. We show that all elements in R' satisfy the (SYS) conditions in P', by rule induction on R'.

For the base, either the element is in R or is an identity. In both cases the result is clear. For the inductive step, consider the rule:

$$\frac{(q_1, S_1, \sigma_1, q_2, S_2) \in R' \qquad (q_2, S_2, \sigma_2, q_3, S_3) \in R'}{(q_1, S_1, \sigma_1; \sigma_2, q_3, S_3) \in R'}\ (\text{TR})$$

and assume that the premises satisfy the (SYS) conditions in P'. Let us write σ for $\sigma_1; \sigma_2$. Suppose $q_1 \xrightarrow{t,i} q_1'$.

- If $i \in \mathsf{dom}(\sigma_1)$ and $j = \sigma_1(i) \in \mathsf{dom}(\sigma_2)$ then $q_2 \xrightarrow{t,j} q_2'$ with $(q_1', S_1)\, P'_{\sigma_1}\, (q_2', S_2)$, and $q_3 \xrightarrow{t,k} q_3'$ with $(q_2', S_2)\, P'_{\sigma_2}\, (q_3', S_3)$ and $k = \sigma'(i) = \sigma_2(j)$. By (TR) we obtain $(q_1', S_1)\, P'_{\sigma}\, (q_3', S_3)$.
- If $i \in \mathsf{dom}(\sigma_1)$ and $j = \sigma_1(i) \notin \mathsf{dom}(\sigma_2)$ then $q_2 \xrightarrow{t,j} q_2'$ with $(q_1', S_1)\, P'_{\sigma_1}\, (q_2', S_2)$, and $q_3 \xrightarrow{t,k^\bullet} q_3'$ with $(q_2', S_2)\, P'_{\sigma_2[j \mapsto k]}\, (q_3', S_3[k])$ for some k. By (TR) we obtain $(q_1', S_1)\, P'_{\sigma[i \mapsto k]}\, (q_3', S_3[k])$.
- If $i \notin \mathsf{dom}(\sigma_1)$ then $q_2 \xrightarrow{t,j^\bullet} q_2'$ with $(q_1', S_1)\, P'_{\sigma_1[i \mapsto j]}\, (q_2', S_2[j])$, for some j, so $q_3 \xrightarrow{t,k^\bullet} q_3'$ with $(q_2', S_2[j])\, P'_{\sigma_2[j \mapsto k]}\, (q_3', S_3[k])$ for some k. By (TR,EXT), using $\sigma_1[i \mapsto j]; \sigma_2[j \mapsto k] \leq_{S_1, S_3[k]} \sigma[i \mapsto k]$, we get $(q_1', S_1)\, P'_{\sigma[i \mapsto k]}\, (q_3', S_3[k])$.

Now suppose $q_1 \xrightarrow{t,i^\bullet} q_1'$.

- Then, $q_2 \xrightarrow{t,j^\bullet} q_2'$ with $(q_1', S_1[i])\, P'_{\sigma_1[i \mapsto j]}\, (q_2', S_2[j])$, for some j, so $q_3 \xrightarrow{t,k^\bullet} q_3'$ with $(q_2', S_2[j])\, P'_{\sigma_2[j \mapsto k]}\, (q_3', S_3[k])$ for some k. By (TR,EXT), $(q_1', S_1[i])\, P'_{\sigma[i \mapsto k]}\, (q_3', S_3[k])$.
- If $k \in \mathsf{rng}(\sigma_2)$ and $j = \sigma_2^{-1}(k) \notin \mathsf{rng}(\sigma_1)$ then $q_2 \xrightarrow{t,j} q_2'$ with $(q_1', S_1[i])\, P'_{\sigma_1[i \mapsto j]}\, (q_2', S_2)$, and $q_3 \xrightarrow{t,k} q_3'$ with $(q_2', S_2)\, P'_{\sigma_2}\, (q_3', S_3)$. By (TR) obtain $(q_1', S_1[i])\, P'_{\sigma[i \mapsto k]}\, (q_3', S_3)$.
- If $k \in S_3 \setminus \mathsf{rng}(\sigma_2)$ then $q_2 \xrightarrow{t,j^\bullet} q_2'$ with $(q_1', S_1[i])\, P'_{\sigma_1[i \mapsto j]}\, (q_2', S_2[j])$, for some j, and so $q_3 \xrightarrow{t,k} q_3'$ with $(q_2', S_2[j])\, P'_{\sigma_2[j \mapsto k]}\, (q_3', S_3)$. By (TR,EXT) we obtain $(q_1', S_1[i])\, P'_{\sigma[i \mapsto k]}\, (q_3', S_3)$.

Consider now the rule:

$$\frac{(q_1, S_1, \sigma, q_2, S_2) \in R' \qquad \sigma \leq_{S_1, S_2} \sigma'}{(q_1, S_1, \sigma', q_2, S_2) \in R'}\ (\textsc{Ext})$$

and assume $(q_1, S_1, \sigma, q_2, S_2)$ satisfies the (SYS) conditions in P'. Suppose $q_1 \xrightarrow{t,i} q_1'$.

- If $i \in \mathsf{dom}(\sigma)$ then $q_2 \xrightarrow{t,\sigma(i)} q_2'$ and $(q_1', S_1)\, P'_{\sigma}\, (q_2', S_2)$. Since $\sigma \subseteq \sigma'$, we have $\sigma(i) = \sigma'(i)$ and $(q_1', S_1)\, P'_{\sigma'}\, (q_2', S_2)$.
- If $i \notin \mathsf{dom}(\sigma')$ then also $i \notin \mathsf{dom}(\sigma)$ and therefore $q_2 \xrightarrow{t,j^\bullet} q_2'$, for some j, and $(q_1', S_1)\, P'_{\sigma[i \mapsto j]}\, (q_2', S_2[j])$. From $\sigma \leq_{S_1, S_2} \sigma'$ we obtain $\sigma[i \mapsto j] \leq_{S_1, S_2[j]} \sigma'[i \mapsto j]$, so $(q_1', S_1)\, P'_{\sigma'[i \mapsto j]}\, (q_2', S_2[j])$.
- If $i \in \mathsf{dom}(\sigma') \setminus \mathsf{dom}(\sigma)$ then we reason as follows. Let $\sigma'(i) = j \in S_2$.
  - I. Since $i \notin \mathsf{dom}(\sigma)$, there is some $q_2 \xrightarrow{t,j'^\bullet} q_2''$ with $(q_1', S_1)\, P'_{\sigma[i \mapsto j']}\, (q_2'', S_2[j'])$;
  - II. hence, there is some $q_1 \xrightarrow{t,i'^\bullet} q_1''$ with $(q_1'', S_1[i'])\, P'_{\sigma[i' \mapsto j']}\, (q_2'', S_2[j'])$;
  - III. then, there is some $q_2 \xrightarrow{t,j} q_2'$ with $(q_1'', S_1[i'])\, P'_{\sigma[i' \mapsto j]}\, (q_2', S_2)$.

  Taking stock (and using symmetry of P'),

$$(q_1', S_1)\, P'_{\sigma[i \mapsto j']}\, (q_2'', S_2[j'])\, P'_{\sigma^{-1}[j' \mapsto i']}\, (q_1'', S_1[i'])\, P'_{\sigma[i' \mapsto j]}\, (q_2', S_2)$$

  and thus, since $\sigma[i \mapsto j']; \sigma^{-1}[j' \mapsto i']; \sigma[i' \mapsto j] \leq_{S_1, S_2} \sigma[i \mapsto j]$, we have $(q_1', S_1)\, P'_{\sigma[i \mapsto j]}\, (q_2', S_2)$.

Suppose now $q_1 \xrightarrow{t,i^\bullet} q_1'$.

- Then, $q_2 \xrightarrow{t,j^\bullet} q_2'$ and $(q_1', S_1[i])\, P'_{\sigma[i \mapsto j]}\, (q_2', S_2[j])$. Since $\sigma[i \mapsto j] \leq_{S_1[i], S_2[j]} \sigma'[i \mapsto j]$, we have $(q_1', S_1[i])\, P'_{\sigma'[i \mapsto j]}\, (q_2', S_2[j])$.
- If $j \in S_2 \setminus \mathsf{rng}(\sigma')$ then $j \notin \mathsf{rng}(\sigma)$, hence $q_2 \xrightarrow{t,j} q_2'$ and $(q_1', S_1[i])\, P'_{\sigma[i \mapsto j]}\, (q_2', S_2)$. Again, we obtain $(q_1', S_1[i])\, P'_{\sigma'[i \mapsto j]}\, (q_2', S_2)$.

Hence, all elements of R' satisfy the (SYS) conditions in P'. □

C. Proof of Lemma 17

Lemma 61. *Suppose* $(p, S)\ R_\sigma\ (q, S)$. *Then* $\mathsf{dom}(\sigma) \supseteq X_S^p(R)$ *and* $\mathsf{rng}(\sigma) \supseteq X_S^q(R)$.

Proof: Since R is closed, $(p, S)\ R_{\sigma;\sigma^{-1}}\ (p, S)$. Because $\sigma; \sigma^{-1} = \mathsf{id}_X$ for some $X \subseteq S$, we have $X \in \mathcal{J}_Q^p(R)$ and, thus $X \supseteq X_S^p(R)$. Because $\sigma; \sigma^{-1} = \mathsf{id}_X$, we must have $\mathsf{dom}(\sigma) \supseteq X$. Hence $\mathsf{dom}(\sigma) \supseteq X_S^p(R)$.

A symmetric argument establishes that $\mathsf{dom}(\sigma^{-1}) \supseteq X_S^q(R)$. Since $\mathsf{rng}(\sigma) = \mathsf{dom}(\sigma^{-1})$, the lemma follows. ■

Lemma 62. *Given* $(p, S) \overset{\mathsf{s}}{\sim}_\sigma (q, S)$, *consider* $\sigma' = \sigma \cap (X_S^p(R) \times X_S^q(R))$. *Then* $\mathsf{dom}(\sigma') = X_S^p(R)$ *and* $\mathsf{rng}(\sigma) = X_S^q(R)$, *i.e.* σ' *is a bijection between* $X_S^p(R)$ *and* $X_S^q(R)$.

Proof: Observe that $\sigma' = \mathsf{id}_{X_S^p(R)}; \sigma; \mathsf{id}_{X_S^q(R)}$. By the preceding lemma, $\mathsf{dom}(\sigma') \supseteq X_S^p(R)$ and $\mathsf{rng}(\sigma') \supseteq X_S^q(R)$. On the other hand, because of $\mathsf{id}_{X_S^p(R)}$ and $\mathsf{id}_{X_S^q(R)}$ in the definition of σ', we have $\mathsf{dom}(\sigma') \subseteq X_S^p(R)$ and $\mathsf{rng}(\sigma') \subseteq X_S^q(R)$. Hence, $\mathsf{dom}(\sigma') = X_S^p(R)$ and $\mathsf{rng}(\sigma') = X_S^q(R)$. Because σ' is injective (as a composite of injections), it must be a bijection between $X_S^p(R)$ and $X_S^q(R)$. ■

The above Lemma shows that $(R \restriction S)$ can be generated from elements of the form $(p, S)\ R_\sigma\ (q, S)$, where σ is a bijection between $X_S^p(R)$ and $X_S^q(R)$, using up-closure under $\leq_{S,S}$. That is, $(p, S)\ R_{\sigma'}\ (q, S)$ iff there exists a bijection $\sigma : X_S^p(R) \to X_S^q(R)$ such that $\sigma \leq_{S,S} \sigma'$ and $(p, S)\ R_\sigma\ (q, S)$.

We can now prove Lemma 17.

Proof: First, since $(p, S)\ R_{\mathsf{id}_S}\ (p, S)$, we have $\mathsf{id}_{X_S^p} \in \mathcal{G}_S^p$. Now, $(p, S)\ R_\sigma\ (p, S)$ implies $(p, S)\ R_{\sigma^{-1}}\ (p, S)$. By the Lemma 62, $\sigma \cap (X_S^p(R) \times X_S^p(R))$ and $\sigma^{-1} \cap (X_S^p(R) \times X_S^p(R))$ are bijections. Thus, $(\sigma \cap (X_S^p(R) \times X_S^p(R))); (\sigma^{-1} \cap (X_S^p(R) \times X_S^p(R))) = \mathsf{id}_{X_S^p(R)}$. ■

D. Proof of Lemma 18

Before we come to the proof of the main result, we shall need the following theorem of Babai which concerns subgroup chains in a group G:

$$G = G_0 > G_1 > \cdots > G_m = I$$

in which I is the trivial 1-point identity group and, for all $i \in [0, m-1]$, G_{i+1} is a subgroup of G_i.

Theorem 63. *([4]) For $n \geq 2$, the length of every subgroup chain in $\mathcal{S}_{[1,n]}$ is at most $2n-3$.*

Now we come to proof of Lemma 18 from the main text.

Proof: Fix $S_1, S_2 \subseteq [1,r]$. We argue that $\{\overset{i}{\sim} \mid (\overset{i+1}{\sim} \cap \mathcal{U}_{S_1,S_2}) \subsetneq (\overset{i}{\sim} \cap \mathcal{U}_{S_1,S_2})\}$ has length at most $4|Q|^2 + 4r^2|Q| - 2r|Q|$.

Let us say that two configurations (q_1, S_1') and (q_1, S_2') are *separated* in $\overset{i}{\sim}$ just if there is no σ such that $(q_1, S_1') \overset{i}{\sim}_\sigma (q_2, S_2')$, we say they are *unseparated* otherwise. We claim that if $(\overset{i+1}{\sim} \cap \mathcal{U}_{S_1,S_2}) \subsetneq (\overset{i}{\sim} \cap \mathcal{U}_{S_1,S_2})$ then there is some $q \in Q$ and $S \in \{S_1, S_2\}$ such that either:

(i) $X_S^q(\overset{i+1}{\sim}) > X_S^q(\overset{i}{\sim})$
(ii) or $\mathcal{G}_S^q(\overset{i+1}{\sim})$ is a strict subgroup of $\mathcal{G}_S^q(\overset{i}{\sim})$
(iii) or there are configurations (q_1, S_1') and (q_2, S_2') ($\{S_1', S_2'\} \subseteq \{S_1, S_2\}$) that are unseparated in $\overset{i}{\sim}$ and become separated in $\overset{i+1}{\sim}$.

We argue as follows. If $(\overset{i+1}{\sim} \cap \mathcal{U}_{S_1,S_2}) \subsetneq (\overset{i}{\sim} \cap \mathcal{U}_{S_1,S_2})$ then there is some $p, q \in Q$, $S_1', S_2' \in \{S_1, S_2\}$ and σ such that $(q_1, S_1') \overset{i}{\sim}_\sigma (q_2, S_2')$ but not $(q_1, S_1') \overset{i+1}{\sim}_\sigma (q_2, S_2')$. Note that, in such a case it follows that also $(q_1, S_1') \overset{i}{\sim}_{\sigma'} (q_2, S_2')$ and $(q_1, S_1') \overset{i+1}{\not\sim}_{\sigma'} (q_2, S_2')$, where $\sigma' = \sigma \cap (X_{S_1'}^{q_1}(\overset{i}{\sim}) \times X_{S_2'}^{q_2}(\overset{i}{\sim}))$, by composing with partial identites. Hence, we assume wlog that $\mathsf{dom}(\sigma) = X_{S_1'}^{q_1}(\overset{i}{\sim})$ and $\mathsf{rng}(\sigma) = X_{S_2'}^{q_2}(\overset{i}{\sim})$. Now, assume that, for all $q \in Q$, $S \in \{S_1, S_2\}$, $X_S^q(\overset{i+1}{\sim}) = X_S^q(\overset{i}{\sim})$ and no previously unseparated pair of configurations become separated in $\overset{i+1}{\sim} \cap \mathcal{U}_{S_1,S_2}$. It follows that there is some τ such that $(q_1, S_1') \overset{i+1}{\sim}_\tau (q_2, S_2')$ and hence $\sigma; \tau^{-1} \in \mathcal{G}_{S_1'}^{q_1}(\overset{i}{\sim})$ but $\sigma; \tau^{-1} \notin \mathcal{G}_{S_1'}^{q_1}(\overset{i+1}{\sim})$ so that $\mathcal{G}_{S_1'}^{q_1}(\overset{i}{\sim}) > \mathcal{G}_{S_1'}^{q_1}(\overset{i+1}{\sim})$.

Since (i) may happen at most $2|Q|r$ times and (iii) may happen at most $4|Q|^2$ times and, by Theorem 63, (ii) may happen at most $2r-2$ times (we relax the bound given by the theorem slightly so as to include the case $r = 1$) for each group, of which there are r possible groups per pair (q, S) since there is a subgroup chain associated with each set $X_S^q(\overset{i}{\sim})$. Hence, this gives an overall bound on the length of the chain $(\overset{i}{\sim} \cap \mathcal{U}_{S_1,S_2})_{i \in I}$ of $4|Q|^2 + 4r^2|Q| - 2r|Q|$. ∎

E. Proof of Proposition 22

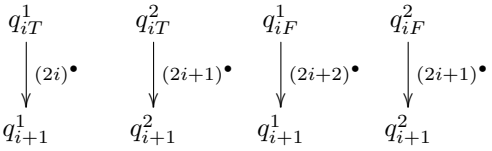
We reduce from TQBF, i.e. the problem of deciding whether a formula Φ of the shape $\Box_1 x_1 \cdots \Box_h x_h . \phi(x_0, \cdots, x_h)$ (with ϕ in conjunctive normal form) is true.

We shall construct a $(2h+1)$-register RA($S\#_0$) and configurations (κ^1, κ^2) such that $\kappa^1 \sim \kappa^2$ if and only if Φ is true. For $j \in \{1, 2\}$, we shall have $\kappa^j = (init^j, \tau^j)$ with $\tau^j(i) = \#$ for all i. The initial transitions will simply initialize the first register, whose content will never change and will be used in Attacker/Defender forcing widgets so that they do not interfere with other registers.

Registers $2, \cdots, 2h+1$ will represent truth-value assignments. Registers $2i, 2i+1$ will be used to represent the value of x_i ($i = 1, \cdots, h$). The values will be selected by Attacker (when $\Box_i = \forall_i$) or Defender (when $\Box_i = \exists$) and we shall use the forcing circuits AF and DF to that end. More precisely, for $i = 1, \cdots, h$, we use $AF(q_i^1, q_i^2, q_{iT}^1, q_{iT}^2, q_{iF}^1, q_{iF}^2)$ or $DF(q_i^1, q_i^2, q_{iT}^1, q_{iT}^2, q_{iF}^1, q_{iF}^2)$ and follow up the state choices with initializations of registers subject to the following conditions:

- register $2i$ is filled if and only if the value of x_i is true,
- register $2i+1$ is filled if and only if the value of x_i is false.

Formally, we add the following transitions.

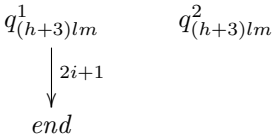


The above handles quantification. To represent the formula $\phi = \phi_1 \wedge \cdots \wedge \phi_k$, we iterate the AF circuit $(k-1)$ times so that Attacker can force the play from (q_{h+1}^1, q_{h+1}^2) into any of $(q_{(h+2)l}^1, q_{(h+2)l}^2)$ for $l = 1, \cdots, k$.

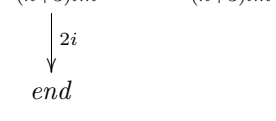
Now assume $\phi_l = \phi_{l1} \vee \cdots \vee \phi_{ln_l}$, where $\phi_{lm} = X_i$ or $\phi_{lm} = \neg X_i$ ($m = 1, \cdots, n_l$). To represent ϕ_l, we iterate the DF circuit $n_l - 1$ times so that Defender can force the play from $(q_{(h+2)l}^1, q_{(h+2)l}^2)$ into any of $(q_{(h+3)lm}^1, q_{(h+3)lm}^2)$ for $m = 1, \cdots, n_l$.

Finally, we need to handle the formulas ϕ_{lm}.

- If $\phi_{lm} = X_i$ we add



- If $\phi_{lm} = \neg X_i$ we add



Note that the outgoing transitions are added only for states subscripted with 1. They give Attacker a chance to win if ϕ_{lm} does *not* hold after Defender's choices.

Overall the construction yields a winning strategy for Defender if and only if the given formula is true. □

F. Proof of Lemma 23

Proof: We view $\mathcal{A}_1, \mathcal{A}_2$ as r-RA($S\#_0$)s with some unused registers and consider the r-RA($S\#_0$) $\mathcal{A} = \langle Q_1 \uplus Q_2 \uplus \{q_0\} \uplus \{q_s\}, q_0, \{(i,\#) \mid i \in [1,r]\}, \delta_1 \cup \delta_2 \cup \delta_s \cup \delta'_s \cup \delta_F, \emptyset\rangle$, where q_0 is a "blind" initial state, q_s is a sink state, $\delta_s = \{q \xrightarrow{t,i} q_s \mid \delta(q) \upharpoonright (t,i) = \emptyset\} \cup \{q \xrightarrow{t,1^\bullet} q_s \mid \delta(q) \upharpoonright (t,i^\bullet) = \emptyset\}$ adds any missing outgoing transitions to $\delta = \delta_1 \cup \delta_2$, $\delta'_s = \{q_s \xrightarrow{t,i} q_s \mid t \in \Sigma \wedge i \in [1,r]\} \cup \{q_s \xrightarrow{t,1^\bullet} q_s \mid t \in \Sigma\}$ is a set of sink transitions, and $\delta_F = \{q \xrightarrow{t_F,i} q_0 \mid i \in [1,r] \wedge q \in F_1 \cup F_1\} \cup \{q \xrightarrow{t_F,1^\bullet} q_0 \mid q \in F_1 \cup F_2\}$ is a set of "final" transitions for some newly introduced constant t_F.

Assume WLOG that $\mathcal{L}(\mathcal{A}_1) \not\subseteq \mathcal{L}(\mathcal{A}_2)$. Then, there is some transition path for $\mathcal{A}_1$ from (q_{01}, ρ_{01}) to some $q_1 \in F_1$ that, when simulated by $\mathcal{A}_2$ from (q_{02}, ρ_{02}), does not lead in F_2. For $\mathcal{A}$, this means that (q_{01}, ρ_{01}) and (q_{02}, ρ_{02}) are not bisimilar: Attacker can lead the game to a configuration pair $((q_1, \rho_1), (q_2, \rho_2))$, with $q_2 \in (Q_2 \setminus F_2) \cup \{q_s\}$, where he wins by playing some (t_F, a) from (q_1, ρ_1). By Proposition 20, Attacker has some strategy $\mathcal{T}$ of depth $O(rN^2 + r^3N)$ for winning the same game. We observe that, because $\mathcal{A}$ is saturated with sink transitions, the latter can only be achieved by Attacker being able to play a final transition with label (t_F, a) in one part of the game. Suppose the happens in the part starting from (q_{01}, ρ_{01}) and let $w\,(t_F, a)$ be the string accepted by the corresponding transition path, so $w \in \mathcal{L}(\mathcal{A}_1)$. By determinacy of $\mathcal{A}_2$, $w \notin \mathcal{L}(\mathcal{A}_2)$. ■

APPENDIX D
PSPACE COMPLETENESS OF INVERSE SUBSEMIGROUP MEMBERSHIP

Given an inverse semigroup $\mathcal{G}$, an *inverse subsemigroup* of $\mathcal{G}$ is some inverse semigroup $\mathcal{H} \subseteq \mathcal{G}$. The problem of inverse subsemigroup membership of $\mathcal{G}$:

> For a set G of elements of $\mathcal{G}$ and a distinguished element g of $\mathcal{G}$, does $g \in \langle G \rangle$?

where $\langle G \rangle$ is the inverse semigroup generated by the members of G via composition and inversion. In this section we prove the following result.

Theorem 64. *Checking membership in inverse subsemirgoups of $\mathcal{IS}_n$ is PSPACE-complete.*

Note first that PSPACE membership follows from Kozen's corresponding PSPACE result for functions, as members of $\mathcal{IS}_n$ can be seen as functions on $[1,n] \cup \{\#\}$.

Theorem 65 (Kozen [13])**.** *Checking whether a function $h : [1,n] \to [1,n]$ can be generated from given functions $f_1, \cdots, f_k : [1,n] \to [1,n]$ is PSPACE-complete.*

For hardness, we shall make use of a result of Lewis and Papadimitriou which shows that PSPACE computations correspond to computations performed in polynomial space by Turing machines with symmetric transitions.

Definition 66. [Lewis & Papadimitriou[9]]A *symmetric Turing Machine* is a tuple $\mathcal{M} = \langle Q, q_0, \delta, F\rangle$ where:

- Q is a set of states, $q_0 \in Q$ is initial and $F \subseteq Q$ is final,
- $\delta \subseteq (Q \times \{0,1\} \times \{0\} \times \{0,1\} \times Q) \cup (Q \times \{0,1\}^2 \times \{-1,+1\} \times \{0,1\}^2 \times Q)$ is the transition relation,

such that $\delta = \delta^{-1}$, where $\delta^{-1} = \{t^{-1} \mid t \in \delta\}$ and:

- $(q,a,0,b,q')^{-1} = (q',b,0,a,q)$,
- $(q,a,b,A,c,d,q')^{-1} = (q',c,d,-A,a,b,q)$.

Note that our machines have input and tape alphabet $\{0,1\}$. Moreover, since we are only examining machines running in polynomial space, we assume a single tape (i.e. no separate input/work tapes), which is initially empty.[10] A symmetric TM $\mathcal{M}$ operates just as a TM, with the feature that $\mathcal{M}$ can look 2 symbols ahead:[11] e.g. a transition $(q,a,b,+1,c,d,q')$ means that, if the automaton is at state q, with the tape symbol at the head being a and the tape symbol to the right of the head being b, then the automaton will rewrite those symbols to c,d respectively, move the head to the right and go to state q'. In a transition $(q,a,b,-1,c,d,q')$ we have the dual behaviour: the automaton looks one symbol to the left ahead, and moves the head to the left. Transitions of the form $(q,a,0,b,q')$ leave the head unmoved.

Given $f : \mathbb{N} \to \mathbb{N}$, we let SSPACE($f$) be the class of problems decided by a symmetric TM in space $O(f)$.

Theorem 67 (Lewis & Papadimitriou[9])**.** *For any $f : \mathbb{N} \to \mathbb{N}$,*

$$\mathrm{DSPACE}(f) \subseteq \mathrm{SSPACE}(f) \subseteq \mathrm{NSPACE}(f).$$

Hence, setting $\mathrm{SPSPACE} = \bigcup_{i \in \mathbb{N}} \mathrm{SSPACE}(n^i)$, using also Savitch's theorem we have $\mathrm{SPSPACE} = \mathrm{PSPACE}$.

Proof of Theorem 64: It suffices to show that the problem is PSPACE-hard. Suppose that $\mathcal{M}$ is a symmetric TM with set of states $Q = [1,K]$ and a tape of size N. By convention, we assume that the initial state is 1, the initial head position is 1 and the unique final state is K. We will simulate its computation using partial permutations from $\mathcal{IS}_n$, where $n = 2N + N + K + 1$.

The first $2N$ numbers in n are used for modelling the tape, the next N numbers for storing the position of the head on the tape, and the last $K+1$ ones for storing the current state, where we include an extra dummy state $(K+1)$ to be used at the beginning of the simulation. The way we model these data (tape, head, state) is by employing $N+1+1$ "tokens" which we distribute among our n numbers as follows:

- One token is shared between $2i-1$ and $2i$, for each $i \in [1,N]$. This token represents the value of bit i of the tape. E.g. if the tape is $10\cdots0$, then we can think of the tokens being on numbers $2,3,5,\cdots,2N-1$.

[9]Harry R. Lewis and Christos H. Papadimitriou. Symmetric Space-Bounded Computation. *Theoretical Computer Science*, 19:161–187, 1982.

[10]Lewis & Papadimitriou work with multi-tape automata, which they reduce to 2-tape automata with one tape for input and one work tape. The same procedure can be used to reduce to just one tape, retaining the same space complexity if the initial complexity is at least polynomial.

[11]This feature does not add expressiveness to a TM but allows one to define symmetric machines.

- One token is shared between the numbers $2N+1, \cdots, 3N$. This token represents the position of the head. E.g. if the tape is on position 5, then this token will be on number $2N+5$.
- One token is shared between the numbers $3N+1, ..., 3N+K+1$. This token represents the current state.

Initially, we will require all tokens to be on positions $2i-1$ ($i \in [1,N]$), $2N+1$ and $3N+K+1$. The latter means that the last token is initially placed on the dummy state $K+1$.

We model transitions as partial permutations that pass on the $2N+2$ tokens. E.g. consider the transition $t = (3,0,0,+1,1,0,5)$.[12] Then, t is modelled by partial permutations:

$$\begin{aligned}\pi_t^i = &\{(2i-1,2i)\} \cup \{(2(i+1)-1, 2(i+1)-1)\} \\ &\cup \{(j,j) \in [1,2N] \times [1,2N] \mid j \neq 2i, 2i-1\} \\ &\cup \ \{(2N+i, 2N+i+1)\} \\ &\cup \ \{(3N+3, 3N+5)\}\end{aligned}$$

for $i \in [1, N-1]$. The first line above says "at position i, read 0 and write 1"; the second line "at position $i+1$, read 0 and write 0"; third line "move right"; and the fourth one "from state 3 go to state 5". This can be generalised to all of δ:

- for all $t = (x,a,b,A,c,d,y)$ and $i \in [1,N]$ such that $i+A \in [1,N]$, set $\pi_t^i = \{(2i-2+A+a, 2i-2+A+c)\} \cup \{(2i+A+b, 2i+A+d)\} \cup \{(j,j) \in [1,2N] \times [1,2N] \mid j \notin [2i-2+A, 2i+1+A]\} \cup \{(2N+i, 2N+i+A)\} \cup \{(3N+x, 3N+y)\}$
- for all $t = (x,a,0,b,y)$ and $i \in [1,N]$, set $\pi_t^i = \{(2i-1+a, 2i-1+b)\} \cup \{(j,j) \in [1,2N] \times [1,2N] \mid j \notin [2i-1, 2i]\} \cup \{(2N+i, 2N+i)\} \cup \{(3N+x, 3N+y)\}$

Note that, in the latter case, $(\pi_t^i)^{-1} = \pi_{t^{-1}}^i$ and, in the former one, $(\pi_t^i)^{-1} = \pi_{t^{-1}}^{i+A}$.

Let us write X for the set of all such partial permutations. If $\mathcal{M}$ has d many transitions then the size of X is at most $d \cdot N$. Let us also select Y to be a minimal set of generators for the group of partial permutations of the form:

$$\pi' = \pi_1 \cup \pi_2 \cup \{(3N+K, 3N+K)\}$$

where $\pi_1 : [1,2N] \xrightarrow{\cong} [1,2N]$ and $\pi_2 : [2N+1, 3N] \xrightarrow{\cong} [2N+1, 3N]$. Note that $|Y| \leq 3n/2$. Moreover, let us take

$$\begin{aligned}\pi_0 = \{(2i-1, 2i-1) \mid i \in [1,N]\} \cup \{(2N+1, 2N+1)\} \\ \cup \{(3N+K+1, 3N+1)\}\end{aligned}$$

to be a permutation setting up the initial positions of the tokens. We then have that:

$$\mathcal{M} \text{ terminates} \iff \pi_{\mathcal{M}} \in \langle X \cup Y \cup \{\pi_0\} \rangle \qquad (1)$$

where $\pi_{\mathcal{M}}$ is the partial permutation:

$$\begin{aligned}\pi_{\mathcal{M}} = \{(2i-1, 2i-1) \mid i \in [1,N]\} \cup \{(2N+1, 2N+1)\} \\ \cup \{(3N+K+1, 3N+K)\}\end{aligned}$$

[12] i.e. from state 3, if the head of the tape and its right-successor read 00 then write 10 to them, move right and go to state 5.

To prove (1), note first that any accepting run of $\mathcal{M}$, say

$$(q_0, H_0, \alpha_0) \xrightarrow{t_1} (q_1, H_1, \alpha_1) \cdots \xrightarrow{t_k} (q_k, H_k, \alpha_k)$$

where $q_0 = 1$, $H_0 = 1$, $\alpha_0 = 0^N$ and $q_k = K$, yields a permutation $\pi = \pi_0; \pi_{t_1}^{H_0}; \cdots; \pi_{t_k}^{H_k}$ with the property that $\mathsf{dom}(\pi) = \mathsf{dom}(\pi_0)$ and $\pi(3N+1) = 3N+K$. We can now select some $\pi' \in \langle Y \rangle$ such that $\pi' \restriction \mathsf{dom}(\pi) = (\pi \restriction [1, 3N]) \cup \{(3N+K, 3N+K)\}$ and, hence, $\pi; \pi'^{-1} = \pi_{\mathcal{M}}$.

Conversely, suppose that $\pi_{\mathcal{M}} \in \langle X \cup Y \cup \{\pi_0\} \rangle$ and in particular let $\pi_{\mathcal{M}} = \pi_0; \pi_1; \cdots; \pi_k$ be a production (so each π^i is in $X \cup Y \cup \{\pi_0\} \cup X^{-1} \cup Y^{-1} \cup \{\pi_0^{-1}\}$). Note that, because π_0 is the only generator with $3N+K+1$ in its domain, it must be the leftmost one in the production. Let $k' \leq k$ be the least index such that $\pi_{k'} \notin Y \cup Y^{-1}$ and, for all $j > k'$, $\pi_j \in Y \cup Y^{-1}$, and assume the production is minimal with respect to the value (k', k) (in the lexicographic ordering). We first claim that there is no π_j with $j < k'$ such that $\pi_j \in Y \cup Y^{-1}$. Because if that were the case then $\pi' = \pi_0; \cdots; \pi_{j-1}$ would satisfy $\mathsf{dom}(\pi') = \mathsf{dom}(\pi_{\mathcal{M}})$ and $\pi'(3N+K+1) = 3N+K$ so there would be some $\pi'' \in \langle Y \rangle$ such that $\pi_{\mathcal{M}} = \pi_0; \cdots; \pi_{j-1}; \pi''$, and the latter would lead to a production with size $(j-1, \cdots)$ which would be smaller than (k', k). Moreover, if $\pi_i = \pi_0$ for some $i > 0$ then we must have $\pi_{i-1} = \pi_0^{-1}$. Because $\pi_0^{-1}; \pi_0 = \mathsf{id}_{\mathsf{rng}(\pi_0)}$ and $|\pi_0^{-1}; \pi_0| = |\pi_{\mathcal{M}}| = N+2$, we have that $\pi_0^{-1}; \pi_0$ can be safely removed from the production of π, thus contradicting the minimality of the latter. For similar reasons, $\pi_i \neq \pi_0^{-1}$, for all $i \in [1,k]$. Hence, π_0 only occurs at the beginning of the production and π_0^{-1} does not occur at all. Summing up, $\pi = \pi_0; \pi_A; \pi_B$ with $\pi_A \in \langle X \rangle$ and $\pi_B \in \langle Y \rangle$. We can now see that π_A represents a computation of $\mathcal{M}$ from 1 to K. ∎

APPENDIX E
PROOFS FROM SECTION VII

A. Proof of Lemma 33

Let $(q_1, S_1, \sigma, q_2, S_2, h)$ and $(q_1, S_1', \sigma', q_2, S_2', h)$ be distinct elements of $\mathsf{symb}(\kappa_1, \kappa_2)$, produced from $\hat{\rho}_i$ and $\hat{\rho}_i'$ respectively (for $i = 1, 2$) and let us assume that $(q_1, S_1', \sigma', q_2, S_2', h) \in \overset{\mathsf{s}}{\sim}$. Take $\sigma_i = \hat{\rho}_i; \hat{\rho}_i'$. By definition, $\sigma_i \restriction [1,r] = \mathsf{id}_{S_i \cap [1,r]}$ and $S_i' = \sigma_i \cdot S_i$. Moreover, we can easily verify that $(q_i, S_i) \overset{\mathsf{s}}{\sim}^h_{\sigma_i} (q_i, S_i')$. Hence, $(q_1, S_1) \overset{\mathsf{s}}{\sim}^h_{\sigma_1} (q_1, S_1') \overset{\mathsf{s}}{\sim}^h_{\sigma'} (q_2, S_2') \overset{\mathsf{s}}{\sim}^h_{\sigma_2^{-1}} (q_2, S_2)$ and, using Proposition 37 (which does not depend on this lemma), we get $(q_1, S_1, \sigma_1; \sigma'; \sigma_2^{-1}, q_2, S_2, h) = (q_1, S_1, \sigma, q_2, S_2, h) \in \overset{\mathsf{s}}{\sim}$.

□

B. Proof of Lemma 36

The second part of the lemma is just the following two lemmata.

Lemma 68. *For all $i \in \omega$, $\overset{i+1}{\sim} \subseteq \overset{i}{\sim}$.*

Proof: The proof is by induction on i. When i is 0, the result is trivial as $\overset{i}{\sim}$ is the universe. The inductive step follows from the IH and the fact that if some g satisfies the (FSYS) conditions in $\overset{i+1}{\sim}$ then it does so in $\overset{i}{\sim}$ as well. ∎

Lemma 69. $\bigcap_{i\in\omega} \overset{i}{\sim} = \overset{\mathsf{s}}{\sim}$

Proof: We start with the $\supseteq$ direction and argue that, for all $i \in \omega$, $\overset{\mathsf{s}}{\sim}$ is a lower bound on $\overset{i}{\sim}$. The proof is by induction on i. When $i = 0$ the result is trivial. When $i = k+1$, assume $(q_1, S_1) \overset{\mathsf{s}}{\sim}{}^{h}_{\tau} (q_2, S_2)$. We wish to show that it and its inverse satisfy the (FSYS) conditions in $\overset{k}{\sim}$. By definition, they satisfy the (FSYS) conditions in $\overset{\mathsf{s}}{\sim}$. Now observe that it follows from the induction hypothesis that $\overset{\mathsf{s}}{\sim} \subseteq \overset{k}{\sim}$, so the tuples satisfy the (FSYS) conditions in $\overset{k}{\sim}$.

For the $\subseteq$ direction, we argue that the left-hand side is a symbolic bisimulation. To see this, assume $(q_1, S_1, \tau, q_2, S_2, h) \in \bigcap_{i\in\omega} \overset{i}{\sim}$ so that $(q_1, S_1, \tau, q_2, S_2, h)$ and its inverse satisfy the (FSYS) conditions in $\overset{i}{\sim}$, for all $i \in \omega$. However, this is just to say that they satisfies the (FSYS) conditions in $\bigcap_{i\in\omega} \overset{i}{\sim}$. ∎

Let $\mathcal{A}$ be an r-FRA($S\#_0$). We show a correspondence between bisimulations and symbolic bisimulations for $\mathcal{A}$ from which the result follows.

bisim → s-bisim. Let R be a bisimulation on $\mathcal{A}$. We claim that the relation $R' \subseteq \mathcal{U}$,

$$P = \bigcup\{\, \mathsf{symb}(\kappa_1, \kappa_2) \mid \kappa_i = (q_i, \rho_i, H_i) \in \mathbb{C} \wedge H_1 = H_2\}$$

is a symbolic bisimulation. For the latter (by symmetry) it suffices to show that P is a symbolic simulation, which reduces to showing the (FSYS) conditions true. So suppose that $(q_1, S_1, \sigma, q_2, S_2) \in P^h$ due to some $(q_1, \rho_1, H)R(q_2, \rho_2, H)$. If $h \leq 2r$ then let $\hat{\rho}_i$ be the $3r$-register assignment of type $S\#_0$ used by symb (for $i = 1, 2$), so $\hat{\rho}_i \restriction [1, r] = \rho_i$, $S_i = \mathsf{dom}(\hat{\rho}_i)$, $\mathsf{rng}(\hat{\rho}_i) = H$ and $\sigma = \hat{\rho}_1; \hat{\rho}_2^{-1}$.

Let $q_1 \xrightarrow{t,i} q_1'$ for some $i \in S_1 \cap [1, r]$. Then, $(q_1, \rho_1, H) \xrightarrow{t,a} (q_1', \rho_1, H)$ with $a = \rho_1(i) \in H$ and, hence, $(q_2, \rho_2, H) \xrightarrow{t,a} (q_2', \rho_2', H)$ with $(q_1', \rho_1, H)R(q_2', \rho_2', H)$.

- If $\sigma(i) \in [1, r]$ then $a = \rho_2(\sigma(i))$ and therefore the above transition is due to some $q_2 \xrightarrow{t,\sigma(i)} q_2'$, and $\rho_2' = \rho_2$. Hence, $(q_1', S_1)P^h_\sigma(q_2', S_2)$.
- If $\sigma(i) = j' \in [r{+}1, 3r]$ then $a = \hat{\rho}_2(j') \notin \mathsf{rng}(\rho_2)$ and the above transition is due to some $q_2 \xrightarrow{t,j^\bullet} q_2'$, and $\rho_2' = \rho_2[j \mapsto a]$. Now, taking $\hat{\rho}_2' = \hat{\rho}_2 \circ (j\, j')$, we have $(q_1', S_1, \hat{\rho}_1; \hat{\rho}_2'^{-1}, q_2', \mathsf{dom}(\hat{\rho}_2')) \in P^h$. Since $\hat{\rho}_1; \hat{\rho}_2'^{-1} = (j\, j') \circ \sigma$ and $\mathsf{dom}(\hat{\rho}_2') = (j\, j') \cdot S_2'$, we obtain $(q_1', S_1)P^h_{(j\, j')\circ\sigma}(q_2', (j\, j') \cdot S_2)$.
- If $i \notin \mathsf{dom}(\sigma)$ then $h = \infty$ and the transition is due to some $q_2 \xrightarrow{t,j^\bullet} q_2'$, and $\rho_2' = \rho_2[j \mapsto a]$. Hence, since $\sigma[i \mapsto j] = \rho_1; (\rho_2[j \mapsto a])^{-1}$ and $\mathsf{dom}(\rho_2') = S_2[j]$, we have $(q_1', S_1)P^h_{\sigma[i\mapsto j]}(q_2', S_2[j])$.

Let $q_1 \xrightarrow{t,i^\bullet} q_1'$. For each $a \in H \setminus \mathsf{rng}(\rho_1)$, $(q_1, \rho_1, H) \xrightarrow{t,a} (q_1', \rho_1', H)$ with $\rho_1' = \rho_1[i \mapsto a]$ and, hence, there is some $(q_2, \rho_2, H) \xrightarrow{t,a} (q_2', \rho_2', H)$ with $(q_1', \rho_1', H)R(q_2', \rho_2', H)$. Now, let $a = \hat{\rho}_1(i')$ for $i' \in S_1 \setminus [1, r]$ (if $h \leq 2r$), and $a = \hat{\rho}_2(j)$ for $j \in S_2 \setminus \mathsf{rng}(\sigma)$ (if $h = \infty$); in the former case, set $\hat{\rho}_1' = \hat{\rho}_1 \circ (i\, i')$.

- If $\sigma(i') \in [1, r]$ then $a = \rho_2(\sigma(i'))$ so the transition above is due to some $q_2 \xrightarrow{t,\sigma(i')} q_2'$ and $\rho_2' = \rho_2$. Thus, $(q_1', \mathsf{dom}(\hat{\rho}_1'))P^h_{\hat{\rho}_1';\hat{\rho}_2^{-1}}(q_2', S_2)$ i.e. $(q_1', (i\, i') \cdot S_1)P^h_{\sigma\circ(i\, i')}(q_2', S_2)$
- If $\sigma(i') = j' \in [r{+}1, 3r]$ then $a = \hat{\rho}_2(j') \notin \mathsf{rng}(\rho_2)$ so the transition above is due to some $q_2 \xrightarrow{t,j^\bullet} q_2'$ and $\rho_2' = \rho_2[j \mapsto a]$. Thus, setting $\hat{\rho}_2' = \hat{\rho}_2 \circ (j\, j)$, we obtain $(q_1', \mathsf{dom}(\hat{\rho}_1'))P^h_{\hat{\rho}_1';\hat{\rho}_2'^{-1}}(q_2', \mathsf{dom}(\hat{\rho}_2'))$ i.e. $(q_1', (i\, i') \cdot S_1)P^h_{(j\, j')\circ\sigma\circ(i\, i')}(q_2', (j\, j') \cdot S_2)$
- For $a = \hat{\rho}_2(j)$ with $j \in S_2 \setminus \mathsf{rng}(\sigma)$, the transition is due to some $q_2 \xrightarrow{t,j} q_2'$, and $\rho_2' = \rho_2$. We moreover have $(q_1', S_1[i])P^h_{\sigma[i\mapsto j]}(q_2', S_2)$.

Finally, let $q_1 \xrightarrow{t,\ell_i} q_1'$ with $\ell_i \in \{i^\bullet, i^\circledast\}$. For each $a \notin H$, we have $(q_1, \rho_1, H) \xrightarrow{t,a} (q_1', \rho_1', H')$ with $\rho_1' = \rho_1[i \mapsto a]$ and $H' = H \cup \{a\}$ and, hence, there is some $(q_2, \rho_2, H) \xrightarrow{t,a} (q_2', \rho_2', H')$ with $(q_1', \rho_1', H')R(q_2', \rho_2', H')$. The latter must be due to $q_2 \xrightarrow{t,\ell_j} q_2'$, for some $\ell_j \in \{j^\bullet, j^\circledast\}$, in which case $\rho_2' = \rho_2[j \mapsto a]$.

- If $h < 2r$ then let $\hat{\rho}_1' = \hat{\rho}_1[i' \mapsto a] \circ (i\, i')$ and $\hat{\rho}_2' = \hat{\rho}_2[j' \mapsto a] \circ (j\, j')$, where $i' = \min([r{+}1, 3r] \setminus \mathsf{dom}(\hat{\rho}_1))$ and $j' = \min([r{+}1, 3r] \setminus \mathsf{dom}(\hat{\rho}_2))$. We have $\rho_1' = \hat{\rho}_1' \restriction [1, r]$, similarly for ρ_2', and $\hat{\rho}_1'; \hat{\rho}_2'^{-1} = (j\, j') \circ \sigma[i' \mapsto j'] \circ (i\, i')$, so $(q_1', (i\, i') \cdot S_1)P^{h+1}_{(j\, j')\circ\sigma[i'\mapsto j']\circ(i\, i')}(q_2', (j\, j') \cdot S_2)$.
- If $h = 2r$ then $(q_1', \mathsf{dom}(\rho_1'))P^\infty_{\rho_1';\rho_2'^{-1}}(q_2', \mathsf{dom}(\rho_2'))$. Now observe that $\hat{\rho}_1[i \mapsto a] \restriction [1, r] = \rho_1'$, similarly for ρ_2', and hence $\sigma[i \mapsto j] \cap [1, r]^2 = \rho_1'; \rho_2'^{-1}$.
- If $h = \infty$ then, since $\sigma[i \mapsto j] = \rho_1[i \mapsto a]; (\rho_2[j \mapsto a])^{-1}$, $\mathsf{dom}(\rho_1') = S_1[i]$ and $\mathsf{dom}(\rho_2') = S_2[j]$, we have $(q_1', S_1[i])P^h_{\sigma[i\mapsto j]}(q_2', S_2[j])$. Moreover, if $\ell_i = i^\bullet$ then, since $|H| > |\mathsf{rng}(\rho_1)| + |\mathsf{rng}(\rho_2)|$, there is some $a' \in H \setminus (\mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2))$. We can therefore pick $a = a'$ and the latter would impose $\ell_j = j^\bullet$.

Hence, P is a symbolic bisimulation.

s-bisim → bisim. Let R be a symbolic bisimulation on $\mathcal{A}$ such that, for all pairs of configurations κ_1, κ_2, either $\mathsf{symb}(\kappa_1, \kappa_2) \subseteq R$ or $\mathsf{symb}(\kappa_1, \kappa_2) \cap R = \emptyset$. We claim that the relation

$$R' = \{\, (\kappa_1, \kappa_2) \mid \kappa_i = (q_i, \rho_i, H_i) \wedge H_1 = H_2 \wedge \mathsf{symb}(\kappa_1, \kappa_2) \subseteq R\}$$

is a bisimulation, for which it suffices to show that R' is a simulation. So suppose that $((q_1, \rho_1, H), (q_2, \rho_2, H)) \in R'$ and let $(q_1, S_1, \sigma, q_2, S_2, h) \in \mathsf{symb}((q_1, \rho_1, H), (q_2, \rho_2, H)) \subseteq R$, and if $h \leq 2r$ let $\hat{\rho}_i$ be the $3r$-extension of ρ_i selected by symb. Let $(q_1, \rho_1, H) \xrightarrow{t,a} (q_1', \rho_1', H')$ for some $(t, a) \in \Sigma \times \mathcal{D}$.

If $a \in \mathsf{rng}(\rho_1)$, say $a = \rho_1(i)$, then $q_1 \xrightarrow{t,i} q_1'$ and $\rho_1' = \rho_1$. We distinguish three cases:

- If $a \in \mathsf{rng}(\rho_2)$ then $\sigma(i) \in [1, r]$, so $q_2 \xrightarrow{t,\sigma(i)} q_2'$ and $(q_1', S_1)R^h_\sigma(q_2', S_2)$. Hence, $(q_2, \rho_2, H) \xrightarrow{t,a} (q_2', \rho_2, H)$ and $(q_1', \rho_1, H)R'(q_2', \rho_2, H)$.
- If $a \notin \mathsf{rng}(\rho_2)$ and $h \leq 2r$ then $\sigma(i) = j' \in [r{+}1, 3r]$,

so $q_2 \xrightarrow{t,j^\bullet} q_2'$ and $(q_1', S_1) R^h_{(j\,j')\circ\sigma} (q_2', (j\,j') \cdot S_2)$, for some j. Hence, $(q_2, \rho_2, H) \xrightarrow{t,a} (q_2', \rho_2[j \mapsto a], H)$ and, taking $\hat\rho_2' = \hat\rho_2 \circ (j\,j')$ (so $\hat\rho_2' \restriction [1,r] = \rho_2[j \mapsto a]$), we have $(q_1', \mathsf{dom}(\hat\rho_1)) R^h_{\hat\rho_1;\hat\rho_2'^{-1}} (q_2', \mathsf{dom}(\hat\rho_2'))$ hence $((q_1', \rho_1, H) R' (q_2', \rho_2[j \mapsto a], H)$.

- If $a \notin \mathsf{rng}(\rho_2)$ and $h = \infty$ then $i \in S_1 \setminus \mathsf{dom}(\sigma)$, so $q_2 \xrightarrow{t,j^\bullet} q_2'$ and $(q_1', S_1) R^h_{\sigma[i\mapsto j]} (q_2', S_2[j])$. Hence, $(q_2, \rho_2, H) \xrightarrow{t,a} (q_2', \rho_2[j \mapsto a], H)$ and this $(q_1', \rho_1, H) R' (q_2', \rho_2[j \mapsto a], H)$.

If $a \in H \setminus \mathsf{rng}(\rho_1)$, and either $h \le 2r$ (so $a = \hat\rho_1(i')$ for some $i' > r$) or $h = \infty$ and $a \in \mathsf{rng}(\rho_2)$, then $H' = H$ and there is some $q_1 \xrightarrow{t,i^\bullet} q_1'$ and $\rho_1' = \rho_1[i \mapsto a]$.

- If $h \le 2r$ and $\sigma(i') \in [1,r]$ then $q_2 \xrightarrow{t,\sigma(i')} q_2'$ and $(q_1', (i\,i') \cdot S_1) R^h_{\sigma\circ(i\,i')} (q_2', S_2)$. Thus, since $\rho_2(\sigma(i')) = a$, $(q_2, \rho_2, H) \xrightarrow{t,a} (q_2', \rho_2, H)$ and, via setting $\hat\rho_1' = \hat\rho_1 \circ (i\,i')$, we obtain $(q_1', \rho_1', H) R' (q_2', \rho_2, H)$.
- If $h \le 2r$ and $\sigma(i') = j' \in [r{+}1, r]$ then $q_2 \xrightarrow{t,j^\bullet} q_2'$ with $(q_1', (i\,i') \cdot S_1) R^h_{(j\,j')\circ\sigma\circ(j\,j')} (q_2', (j\,j') \cdot S_2)$, for some j. Thus, since $a \notin \mathsf{rng}(\rho_2)$, $(q_2, \rho_2, H) \xrightarrow{t,a} (q_2', \rho_2[j \mapsto a], H)$ and, via setting $\hat\rho_1' = \hat\rho_1 \circ (i\,i')$ and $\hat\rho_2' = \hat\rho_2 \circ (j\,j')$, we obtain $(q_1', \rho_1', H) R' (q_2', \rho_2[j \mapsto a], H)$.
- If $h = \infty$ and $a \in \mathsf{rng}(\rho_2)$, say $a = \rho_2(j)$, then $j \in S_2 \setminus \mathsf{rng}(\sigma)$. Hence, $q_2 \xrightarrow{t,j} q_2'$ with $(q_1', S_1[i]) R_{\sigma[i\mapsto j]} (q_2', S_2)$, from which we get $(q_2, \rho_2, H) \xrightarrow{t,a} (q_2', \rho_2, H)$ and $(q_1', \rho_1', H) R' (q_2', \rho_2, H)$.

If either $h \le 2r$ and $a \notin H$, or $h = \infty$ and $a \notin \mathsf{rng}(\rho_1) \cup \mathsf{rng}(\rho_2)$ then $q_1 \xrightarrow{t,\ell_i} q_1'$, for some $\ell_i \in \{i^\bullet, i^\circledast\}$, and $H' = H \cup \{a\}$ and $\rho_1' = \rho_i[i \mapsto a]$. Thus, $q_2 \xrightarrow{t,\ell_j} q_2'$ for some $\ell_j \in \{j^\bullet, j^\circledast\}$. Let $\rho_2' = \rho_2[j \mapsto a]$.

- If $h < 2r$ then, taking $i' = \max([r{+}1, 3r] \setminus S_1)$ and $j' = \max([r{+}1, 3r] \setminus S_2)$, we have $(q_1', (i\,i') \cdot S_1) R^{h+1}_{(j\,j')\circ\sigma[i'\mapsto j']\circ(i\,i')} (q_2', (j\,j') \cdot S_2)$. Via setting $\hat\rho_1' = \hat\rho_1[i' \mapsto a] \circ (i\,i')$ and $\hat\rho_2' = \hat\rho_2[j' \mapsto a] \circ (j\,j')$, we obtain $(q_1', \rho_1', H') R' (q_2', \rho_2', H')$.
- If $h = 2r$ then $(q_1', S_1[i] \cap [1,r]) R^\infty_{\sigma[i\mapsto j]\cap[1,r]^2} (q_2', S_2[j] \cap [1,r])$, from which we obtain $(q_1', \rho_1', H') R' (q_2', \rho_2', H')$.
- If $h = \infty$ then $(q_1', S_1[i]) R^h_{\sigma[i\mapsto j]} (q_2', S_2[j])$. In particular, if $a \in H$ then $\ell_i = i^\bullet$ and therefore $\ell_j = j^\bullet$. Thus, in each case, $(q_2, \rho_2, H) \xrightarrow{t,a} (q_2', \rho_2[j \mapsto a], H')$ and $(q_1', \rho_1', H') R' (q_2', \rho_2', H')$.

Hence, R' is a bisimulation.

Thus, to prove Lemma 36, given such κ_1 and κ_2, if $\kappa_1 \overset{\mathsf{s}}{\sim} \kappa_2$ then we can construct a symbolic bisimulation P such that $\mathsf{symb}(\kappa_1, \kappa_2) \subseteq P$. Conversely, if $\kappa_1 \overset{\mathsf{s}}{\sim} \kappa_2$ then, using also Lemma 33, there is a bisimulation R' such that $\kappa_1 R' \kappa_2$. □

C. Proof of Proposition 37

We start with the following lemma.

Lemma 70. *Let $R, P \subseteq \mathcal{U}$ with $R = R^{-1}$. If all $g \in R$ satisfy the* (FSYS) *conditions in P then all $g \in Cl(R)$ satisfy the* (FSYS) *conditions in $Cl(P)$.*

Proof: We first observe that, by symmetry of R, $Cl(R) = Cl^-(R)$ where, for any relation X, we let $Cl^-(X)$ be the smallest relation that contains X and is closed under the rules (ID), (TR) and (EXT). Let us set $\hat R = Cl^-(R)$ and $\hat P = Cl(P)$. We show that all elements of $\hat R$ satisfy the (FSYS) conditions in $\hat P$, by rule induction on $\hat R$.

The base has the element in R or an identity. In both cases the result is clear. For the inductive step, consider the rule:

$$\frac{(q_1, S_1, \sigma_1, q_2, S_2) \in \hat R^h \qquad (q_2, S_2, \sigma_2, q_3, S_3) \in \hat R^h}{(q_1, S_1, \sigma_1; \sigma_2, q_3, S_3) \in \hat R^h} \ (\textsc{Tr})$$

and assume that the premises satisfy the (FSYS) conditions in $\hat P$. Let us write σ for $\sigma_1; \sigma_2$. Suppose $q_1 \xrightarrow{t,i_1} q_1'$.

- If $\sigma_1(i_1) = i_2 \in [1,r]$ then, by the (FSYS) conditions on $(q_1, S_1, \sigma_1, h, q_2, S_2)$, we have $q_2 \xrightarrow{t,i_2} q_2'$ with $j_2 = \sigma_1(j_1)$ and $(q_1', S_1) \hat P^h_{\sigma_1} (q_2', S_2)$.
  - If $\sigma_2(i_2) = i_3 \in [1,r]$ then $q_3 \xrightarrow{t,i_3} q_3'$ with $(q_2', S_2) \hat P^h_{\sigma_2} (q_3', S_3)$. By (TR), $(q_1', S_1) \hat P^h_{\sigma} (q_3', S_3)$.
  - If $\sigma_2(i_2) = i_3' \in [r{+}1, 3r]$ then $q_3 \xrightarrow{t,i_3^\bullet} q_3'$ with $(q_2', S_2) \hat P^h_{(i_3\,i_3')\circ\sigma_2} (q_3', (i_3\,i_3') \cdot S_3)$. By (TR) we obtain $(q_1', S_1) \hat P^h_{(i_3\,i_3')\circ\sigma} (q_3', (i_3\,i_3') \cdot S_3)$, as required.
  - If $i_2 \in S_2 \setminus \mathsf{dom}(\sigma_2)$ then $q_3 \xrightarrow{t,i_3^\bullet} q_3'$ with $(q_2', S_2) \hat P^h_{\sigma_2[i_2\mapsto i_3]} (q_3', S_3[i_3])$. By (TR), we obtain $(q_1', S_1) \hat P^h_{\sigma_1;\sigma_2[i_2\mapsto i_3]} (q_3', S_3[i_3])$, which is what is required since $\sigma[i_1 \mapsto i_3] = \sigma_1; \sigma_2[i_2 \mapsto i_3]$.
- If $\sigma_1(i_1) = i_2' \in [r{+}1, 3r]$ then $q_2 \xrightarrow{t,i_2^\bullet} q_2'$ with $(q_1', S_1) \hat P^h_{(i_2\,i_2')\circ\sigma_1} (q_2', (i_2\,i_2') \cdot S_2)$.
  - If $\sigma_2(i_2') = i_3 \in [1,r]$ then $q_3 \xrightarrow{t,i_3} q_3'$ with $(q_2', (i_2\,i_2') \cdot S_2) \hat P^h_{\sigma_2\circ(i_2\,i_2')} (q_3', S_3)$. By (TR) we obtain $(q_1', S_1) \hat P^h_{\sigma} (q_3', S_3)$.
  - If $\sigma_2(i_2') = i_3' \in [r{+}1, 3r]$ then, from (b2), $q_3 \xrightarrow{t,i_3^\bullet} q_3'$ with $(q_2', (i_2\,i_2') \cdot S_2) \hat P^h_{(i_3\,i_3')\circ\sigma_2\circ(i_2\,i_2')} (q_3', (i_3\,i_3') \cdot S_3)$. By (TR) we have $(q_1', S_1) \hat P^h_{(i_3\,i_3')\circ\sigma} (q_3', (i_3\,i_3') \cdot S_3)$.
- If $i_1 \in S_1 \setminus \mathsf{dom}(\sigma_1)$ then we have $h = \infty$ and $q_2 \xrightarrow{t,i_2^\bullet} q_2'$ with $(q_1', S_1) \hat P^h_{\sigma_1[i_1\mapsto i_2]} (q_2', S_2[i_2])$, for some i_2, so $q_3 \xrightarrow{t,k^\bullet} q_3'$ with $(q_2', S_2[i_2]) \hat P^h_{\sigma_2[i_2\mapsto i_3]} (q_3', S_3[i_3])$ for some i_3. By (TR,EXT), using $\sigma_1[i_1 \mapsto i_2]; \sigma_2[i_2 \mapsto i_3] \le_{S_1, S_3[i_3]} \sigma[i_1 \mapsto i_3]$, we get $(q_1, S_1) \hat P^h_{\sigma[i_1\mapsto i_3]} (q_3, S_3[i_3])$.

Now suppose $q_1 \xrightarrow{t,i_1^\bullet} q_1'$ and let $i_1' \in S_1 \setminus [1,r]$ (so $h \le 2r$).

- If $\sigma_1(i_1') = i_2 \in [1,r]$ then $q_2 \xrightarrow{t,i_2} q_2'$ with $(q_1', (i_1\,i_1') \cdot S_1) \hat P^h_{\sigma_1\circ(i_1\,i_1')} (q_2', S_2)$.
  - If $\sigma_2(i_2) = i_3 \in [1,r]$ then $q_3 \xrightarrow{t,i_3} q_3'$ with $(q_2', S_2) \hat P^h_{\sigma_2} (q_3', S_3)$. By (TR) we obtain $(q_1', (i_1\,i_1') \cdot S_1) \hat P^h_{\sigma\circ(i_1\,i_1')} (q_3', S_3)$.
  - If $\sigma_2(i_2) = i_3' \in [r{+}1, 3r]$ then $q_3 \xrightarrow{t,i_3^\bullet} q_3'$ with $(q_2', S_2) \hat P^h_{(i_3\,i_3')\circ\sigma_2} (q_3', (i_3\,i_3') \cdot S_3)$. By (TR) we have $(q_1', S_1) \hat P^h_{(i_3\,i_3')\circ\sigma} (q_3', (i_3\,i_3') \cdot S_3)$.

- If $\sigma_1(i'_1) = i'_2 \in [r{+}1, 3r]$ then $q_2 \xrightarrow{t,i_2^\bullet} q'_2$ with $(q'_1, (i_1\, i'_1)\cdot S_1)\hat{P}^h_{(i_2\, i'_2)\circ\sigma_1\circ(i_1\, i'_1)}(q'_2, (i_2\, i'_2)\cdot S_2)$.
  - If $\sigma_2(i'_2) = i_3 \in [1, r]$ then $q_3 \xrightarrow{t,i_3} q'_3$ with $(q'_2, (i_2\, i'_2)\cdot S_2)\hat{P}^h_{\sigma_2\circ(i_2\, i'_2)}(q'_3, S_3)$. By (TR) we obtain $(q'_1, (i_1\, i'_1)\cdot S_1)\hat{P}^h_{\sigma\circ(i_1\, i'_1)}(q'_3, S_3)$.
  - If $\sigma_2(i'_2) = i'_3 \in [r{+}1, 3r]$ then $q_3 \xrightarrow{t,i_3^\bullet} q'_3$ with $(q'_2, (i_2\, i'_2)\cdot S_2)\hat{P}^h_{(i_3\, i'_3)\circ\sigma_2\circ(i_2\, i'_2)}(q'_3, (i_3\, i'_3)\cdot S_3)$. By (TR), $(q'_1, (i_1\, i'_1)\cdot S_1)\hat{P}^h_{(i_3\, i'_3)\circ\sigma\circ(i_1\, i'_1)}(q'_3, (i_3\, i'_3)\cdot S_3)$.

On the other hand, if $q_1 \xrightarrow{t,i_1^\bullet} q'_1$ and $i_3 \in S_3 \setminus \mathsf{rng}(\sigma)$ (so $h = \infty$).

- If $i_3 \in \mathsf{rng}(\sigma_2)$ and $i_2 = \sigma_2^{-1}(i_3) \notin \mathsf{rng}(\sigma_1)$ then $q_2 \xrightarrow{t,i_2} q'_2$ with $(q'_1, S_1[i_1])\hat{P}^h_{\sigma_1[i_1\mapsto i_2]}(q'_2, S_2)$, and so $q_3 \xrightarrow{t,i_3} q'_3$ with $(q'_2, S_2)\hat{P}^h_{\sigma_2}(q'_3, S_3)$. By (TR) obtain $(q'_1, S_1[i])\hat{P}^h_{\sigma[i_1\mapsto i_3]}(q'_3, S_3)$.
- If $i_3 \in S_3 \setminus \mathsf{rng}(\sigma_2)$ then, since $q_2 \xrightarrow{t,i_2^\bullet} q'_2$ with $(q'_1, S_1[i_1])\hat{P}^h_{\sigma_1[i_1\mapsto i_2]}(q_2, S_2[i_2])$ for some i_2, we also have $q_3 \xrightarrow{t,i_3} q'_3$ with $(q'_2, S_2[i_2])\hat{P}^h_{\sigma_2[i_2\mapsto i_3]}(q'_3, S_3)$. By (TR,EXT) we obtain $(q'_1, S_1[i_1])\hat{P}^h_{\sigma[i_1\mapsto i_3]}(q'_3, S_3)$.

Finally, let $q_1 \xrightarrow{t,i_1^\bullet/i_1^\circledast} q'_1$. Then, $q_2 \xrightarrow{t,i_2^\bullet/i_2^\circledast} q'_2$ and $q_3 \xrightarrow{t,i_3^\bullet/i_3^\circledast} q'_3$ with $(q'_1, S'_1)\hat{P}^{h'}_{\sigma'_1}(q'_2, S'_2)$ and $(q'_2, S'_2)\hat{P}^{h'}_{\sigma'_2}(q'_3, S'_3)$.

- If $h < 2r$ then $h' = h + 1$ and $i'_k = \min([r{+}1, 3r] \setminus S_k)$, $S'_k = (i_k\, i'_k)\circ S_k[i'_k]$ and $\sigma'_k = (i_{k+1}\, i'_{k+1})\circ(\sigma_k[i'_k \mapsto i'_{k+1}])\circ(i_k\, i'_k)$, for $k = 1, 2, 3$. By (TR), we have $(q'_1, S'_1)\hat{P}^h_{\sigma'_1;\sigma'_2}(q'_3, S'_3)$, which is as required since $\sigma'_1;\sigma'_2 = (i_3\, i'_3)\circ\sigma[i'_1 \mapsto i'_3]\circ(i_1\, i'_1)$.
- If $h = 2r$ then $h' = \infty$ and $S'_k = S_k[i_k] \cap [1, r]$ and $\sigma'_k = \sigma_k[i_k \mapsto i_{k+1}] \cap [1, r]^2$. By (TR), we have $(q'_1, S'_1)\hat{P}^h_{\sigma'_1;\sigma'_2}(q'_3, S'_3)$ and, hence, by (EXT) we obtain the required result since $\sigma'_1;\sigma'_2 \leq_{S'_1,S'_2} \sigma[i_1 \mapsto i_3] \cap [1, r]^2$.
- If $h = \infty$ then $h' = \infty$ and $S'_k = S_k[i_k]$ and $\sigma'_k = \sigma_k[i_k \mapsto i_{k+1}]$. By (TR,EXT), $(q'_1, S_1[i_1])\hat{P}^h_{\sigma[i_1\mapsto i_3]}(q'_3, S_3[i_3])$. Moreover, if the transition from q_1 to q'_1 is localy fresh then so is the one from q_2 to q'_2, and from q_3 to q'_3.

We now consider the rule:

$$\frac{(q_1, S_1, \sigma, q_2, S_2) \in \hat{R}^h \qquad \sigma \leq_{S_1,S_2} \sigma'}{(q_1, S_1, \sigma', q_2, S_2) \in \hat{R}^h}\ (\textsc{Ext})$$

and assume $(q_1, S_1, \sigma, h, q_2, S_2)$ satisfies the (FSYS) conditions in $\hat{P}$. Note that if $h < \infty$ then $h = |\sigma| = |\sigma'|$, hence $\sigma = \sigma'$ and the required result is trivial. So let us assume $h = \infty$. Suppose $q_1 \xrightarrow{t,i} q'_1$.

- If $i \in \mathsf{dom}(\sigma)$ then $q_2 \xrightarrow{t,\sigma(i)} q'_2$ and $(q'_1, S_1)\hat{P}^\infty_\sigma(q'_2, S_2)$. Since $\sigma \subseteq \sigma'$, we have $\sigma(i) = \sigma'(i)$ and $(q'_1, S_1)\hat{P}^\infty_{\sigma'}(q'_2, S_2)$.
- If $i \notin \mathsf{dom}(\sigma')$ then also $i \notin \mathsf{dom}(\sigma)$ and therefore $q_2 \xrightarrow{t,j^\bullet} q'_2$, for some j, and $(q'_1, S_1)\hat{P}^\infty_{\sigma[i\mapsto j]}(q'_2, S_2[j])$. From $\sigma \leq_{S_1,S_2} \sigma'$ we obtain $\sigma[i \mapsto j] \leq_{S_1,S_2[j]} \sigma'[i \mapsto j]$, so $(q'_1, S_1)\hat{P}^\infty_{\sigma'[i\mapsto j]}(q'_2, S_2[j])$.
- If $i \in \mathsf{dom}(\sigma') \setminus \mathsf{dom}(\sigma)$ then we reason as follows. Let $\sigma'(i) = j \in S_2$.
  - I. Since $i \notin \mathsf{dom}(\sigma)$, there is some $q_2 \xrightarrow{t,j'^\bullet} q''_2$ with $(q'_1, S_1)\hat{P}^\infty_{\sigma[i\mapsto j']}(q''_2, S_2[j'])$;
  - II. hence, there is some $q_1 \xrightarrow{t,i'^\bullet} q''_1$ with $(q''_1, S_1[i'])\hat{P}^\infty_{\sigma[i'\mapsto j']}(q''_2, S_2[j'])$;
  - III. then, there is some $q_2 \xrightarrow{t,j} q'_2$ with $(q''_1, S_1[i'])\hat{P}^\infty_{\sigma[i'\mapsto j]}(q'_2, S_2)$.

  Taking stock (and using symmetry of $\hat{P}$),

$$(q'_1, S_1)\hat{P}^\infty_{\sigma[i\mapsto j']}(q''_2, S_2[j'])\hat{P}^\infty_{\sigma^{-1}[j'\mapsto i']}(q''_1, S_1[i'])\hat{P}^\infty_{\sigma[i'\mapsto j]}(q'_2, S_2)$$

  and thus, since $\sigma[i \mapsto j'];\sigma^{-1}[j' \mapsto i';\sigma[i' \mapsto j] \leq_{S_1,S_2} \sigma[i \mapsto j] \leq_{S_1,S_2} \sigma'$, we have $(q'_1, S_1)\hat{P}^\infty_{\sigma'}(q'_2, S_2)$.

Suppose now $q_1 \xrightarrow{t,i^\bullet} q'_1$.

- Then, $q_2 \xrightarrow{t,j^\bullet} q'_2$ and $(q'_1, S_1[i])\hat{P}^\infty_{\sigma[i\mapsto j]}(q'_2, S_2[j])$. Since $\sigma[i \mapsto j] \leq_{S_1[i],S_2[j]} \sigma'[i \mapsto j]$, we have $(q'_1, S_1[i])\hat{P}^\infty_{\sigma'[i\mapsto j]}(q'_2, S_2[j])$.
- If $j \in S_2 \setminus \mathsf{rng}(\sigma')$ then $j \notin \mathsf{rng}(\sigma)$, hence $q_2 \xrightarrow{t,j} q'_2$ and $(q'_1, S_1[i])\hat{P}^\infty_{\sigma[i\mapsto j]}(q'_2, S_2)$. Again, we obtain $(q'_1, S_1[i])\hat{P}^\infty_{\sigma'[i\mapsto j]}(q'_2, S_2)$.

Finally, let $q_1 \xrightarrow{t,i^\circledast} q'_1$.

- Then, $q_2 \xrightarrow{t,j^\circledast} q'_2$ and $(q'_1, S_1[i])\hat{P}^\infty_{\sigma[i\mapsto j]}(q'_2, S_2[j])$. Since $\sigma[i \mapsto j] \leq_{S_1[i],S_2[j]} \sigma'[i \mapsto j]$, we have $(q'_1, S_1[i])\hat{P}^\infty_{\sigma'[i\mapsto j]}(q'_2, S_2[j])$.

Hence, $\hat{R}$ satisfies the (FSYS) conditions in $\hat{P}$. ∎

We now prove Proposition 37.

Proof: For $Cl(\overset{\mathsf{s}}{\sim}) = \overset{\mathsf{s}}{\sim}$, since $\overset{\mathsf{s}}{\sim}$ is symmetric and satisfies the (FSYS) conditions in itself, from the previous lemma we have that $Cl(\overset{\mathsf{s}}{\sim})$ satisfies the (FSYS) conditions in itself and is therefore a symbolic bisimulation. Thus, $Cl(\overset{\mathsf{s}}{\sim}) \subseteq \overset{\mathsf{s}}{\sim}$.

For $Cl(\overset{i}{\sim}) = \overset{i}{\sim}$ we do induction on i. When $i = 0$ then the result follows from the fact that $\overset{0}{\sim}$ is the universal relation. For the inductive case, note first that $\overset{i+1}{\sim}$ is symmetric by construction and all $g \in \overset{i+1}{\sim}$ satisfy the (FSYS) conditions in $\overset{i}{\sim}$. Hence, by Lemma 70, all elements of $Cl(\overset{i+1}{\sim})$ satisfy the (FSYS) conditions in $Cl(\overset{i}{\sim})$. By IH, $Cl(\overset{i}{\sim}) = \overset{i}{\sim}$ so $Cl(\overset{i+1}{\sim}) \subseteq \overset{i+1}{\sim}$, as required. ∎

D. Proof of Lemma 38

More explicitly, the last part of Proposition 37 means that, given $(q_1, S_1) \overset{i}{\sim}^h_\tau (q_2, S_2)$:

1) Then, $(q_2, S_2) \overset{i}{\sim}^h_{\tau^{-1}} (q_1, S_1)$.
2) For all τ', if $\tau \leq_{S_1,S_2} \tau'$ then $(q_1, S_1) \overset{i}{\sim}^h_{\tau'} (q_2, S_2)$.
3) For all $(q_2, S_2) \overset{i}{\sim}^h_{\tau'} (q_3, S_3)$, $(q_1, S_1,) \overset{i}{\sim}^h_{\tau;\tau'} (q_3, S_3)$.

Before we come to the proof of the main result, recall Theorem 63 which says that, for $n \geq 2$, the length of every subgroup chain in $\mathcal{S}_{[1,n]}$ is at most $2n - 3$.

We next prove Lemma 38.

Proof: We argue that the set

$$\{\overset{i}{\sim} \mid (\overset{i+1}{\sim} \cap\, \mathcal{U}^h_{S_1,S_2}) \subsetneq (\overset{i}{\sim} \cap\, \mathcal{U}^h_{S_1,S_2})\}$$

has size at most $4|Q|^2 + 4r^2|Q| - 2r|Q|$.

Let us say that a tuple $(q_1, S_1', h, q_2, S_2')$ is *separated* in $\overset{i}{\sim}$ just if there is no σ such that $(q_1, S_1') \overset{i}{\sim}{}^h_\sigma (q_2, S_2')$; we say it is *unseparated* otherwise. We claim that if $(\overset{i+1}{\sim} \cap\, \mathcal{U}^h_{S_1,S_2}) \subsetneq (\overset{i}{\sim} \cap\, \mathcal{U}^h_{S_1,S_2})$ then there is some $q \in Q$ and $S \in \{S_1, S_2\}$ such that either:

(i) $X^q_S(\overset{i}{\sim}{}^h) \subsetneq X^q_S(\overset{i+1}{\sim}{}^h)$

(ii) or $\mathcal{G}^q_S(\overset{i+1}{\sim}{}^h)$ is a strict subgroup of $\mathcal{G}^q_S(\overset{i}{\sim}{}^h)$

(iii) or there is a tuple $(q_1, S_1', h, q_2, S_2')$ ($\{S_1', S_2'\} \subseteq \{S_1, S_2\}$) that is unseparated in $\overset{i}{\sim}$ and becomes separated in $\overset{i+1}{\sim}$.

We argue as follows. If $(\overset{i+1}{\sim} \cap\, \mathcal{U}^h_{S_1,S_2}) \subsetneq (\overset{i}{\sim} \cap\, \mathcal{U}^h_{S_1,S_2})$ then there is some $p, q \in Q$, $S_1', S_2' \in \{S_1, S_2\}$ and σ such that $(q_1, S_1') \overset{i}{\sim}{}^h_\sigma (q_2, S_2')$ but not $(q_1, S_1') \overset{i+1}{\sim}{}^h_\sigma (q_2, S_2')$. Note that, in such a case it follows that also $(q_1, S_1') \overset{i}{\sim}{}^h_{\sigma'} (q_2, S_2')$ and $(q_1, S_1') \overset{i+1}{\not\sim}{}^h_{\sigma'} (q_2, S_2')$, where $\sigma' = \sigma \cap (X^{q_1}_{S_1'}(\overset{i}{\sim}{}^h) \times X^{q_2}_{S_2'}(\overset{i}{\sim}{}^h))$, by composing with partial identites. Hence, we assume wlog that $\mathsf{dom}(\sigma) = X^{q_1}_{S_1'}(\overset{i}{\sim}{}^h)$ and $\mathsf{rng}(\sigma) = X^{q_2}_{S_2'}(\overset{i}{\sim}{}^h)$. Now, assume that, for all $q \in Q$, $S \in \{S_1, S_2\}$, $X^q_S(\overset{i+1}{\sim}{}^h) = X^q_S(\overset{i}{\sim}{}^h)$ and no previously unseparated tuple becomes separated in $\overset{i+1}{\sim} \cap\, \mathcal{U}^h_{S_1,S_2}$. It follows that there is some τ such that $(q_1, S_1') \overset{i+1}{\sim}{}^h_\tau (q_2, S_2')$ and hence $\sigma; \tau^{-1} \in \mathcal{G}^{q_1}_{S_1'}(h, i)$ but $\sigma; \tau^{-1} \notin \mathcal{G}^{q_1}_{S_1'}(h, i+1)$ so that $\mathcal{G}^{q_1}_{S_1'}(\overset{i}{\sim}{}^h) > \mathcal{G}^{q_1}_{S_1'}(\overset{i+1}{\sim}{}^h)$.

Since (i) may happen at most $4|Q|r$ times and (iii) may happen at most $4|Q|^2$ times and, by Theorem 63, (ii) may happen at most $4r - 4$ times (we relax the bound given by the theorem slightly so as to include the case $r = 1$) for each group, of which there are $2r$ possible groups per pair (q, S) since there is a subgroup chain associated with each set $X^q_S(\overset{i}{\sim}{}^h)$. Hence, this gives an overall bound on the length of the chain $(\overset{i}{\sim} \cap\, \mathcal{U}^h_{S_1,S_2})_{i \in I}$ of $4|Q|^2 + 16r^2|Q| - 12r|Q|$. ∎

E. Proof of Lemma 39

Proof: Part 2 follows from Part 1, by taking the minimum value for $\hat{\gamma}$ ($= 0$).

For 1 we do induction on $(4r - \hat{\gamma}(S_1, S_2, h) + 1)$, starting from the inductive step. Assume the result holds for all (S_1', S_2', h') with $\hat{\gamma}(S_1', S_2', h') > \hat{\gamma}(S_1, S_2, h)$. Let $j' = \hat{c}(4r - (\hat{\gamma}(S_1, S_2, h) + 1) + 2)(|Q|^2 + r^2|Q|) = \hat{c}(4r - \hat{\gamma}(S_1, S_2, h) + 1)(|Q|^2 + r^2|Q|)$. Then, for all such (S_1', S_2', h'), $(\overset{j'}{\sim} \cap\, \mathcal{U}^{h'-}_{S_1',S_2'}) = (\overset{\mathsf{s}}{\sim} \cap\, \mathcal{U}^{h'-}_{S_1',S_2'})$. Now, for $k > j'$, if $\overset{k}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2} = \overset{k+1}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2}$, then we must have $\overset{k}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2} = \overset{\mathsf{s}}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2}$, because the (FSYS) conditions for (S_1, S_2, h) refer to either (S_1, S_2, h) or (S_1', S_2', h') with $\hat{\gamma}(S_1', S_2', h') > \hat{\gamma}(S_1, S_2, h)$. Consequently, if $\overset{j'}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2} \neq \overset{\mathsf{s}}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2}$, the sequence $(\overset{k}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2})$ ($k = j', j'+1, \cdots$) will have to change in every step before stabilisation. Thus, the steps before stabilisation will induce a subchain of the chain in Lemma 38 (2). Hence, at most $\hat{c}(|Q|^2 + r^2|Q|)$ extra steps from $(\overset{j'}{\sim})$ will be required to arrive at $\overset{\mathsf{s}}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2}$, which delivers the required bound.

The base case ($\hat{\gamma}(S_1, S_2, h) = 4r + 1$, i.e. $h = \infty$ and $S_1 = S_2 = [1, r]$) can be established in a similar fashion: in this case the (FSYS) conditions can only refer to (S_1, S_2, h), thus the sequence $(\overset{k}{\sim} \cap\, \mathcal{U}^{h-}_{S_1,S_2})$ ($k \geq 0$) will be strictly decreasing to stabilisation and the bound from Lemma 38 can be applied. ∎

APPENDIX F
PROOF OF THEOREM 46

Definition 71. [[7]] A one-way universal n-register automaton (URA_n) is a tuple $\langle \Sigma, Q, q_I, n, \delta \rangle$ such that Σ is a finite alphabet, Q is a finite set of states, $q_I \in Q$ is the initial state and $\delta : Q \to \Delta(\Sigma, Q, n)$ is the transition function, where

$$\begin{array}{rcl} \Delta(\Sigma, Q, n) & = & \{\, \bot,\ \top,\ q \wedge q',\ q \triangleleft \beta \triangleright q',\ Xq,\ \overline{X}q,\ \downarrow_r q \\ & \mid & \quad q, q' \in Q,\ r \in \{1, \cdots, n\},\ \beta \in B(\Sigma, n)\ \} \\ B(\Sigma, n) & = & \{a, \mathsf{end}\} \cup \{\uparrow_r \mid r \in \{1, \cdots, n\}\} \end{array}$$

The emptiness problem for URA_2 is undecidable [7]. We shall reduce it to bisimilarity testing. We first sketch the argument and then later give all the details.

A. Sketch of argument

Given a URA_2 U, we devise a 2-VPDRA $\mathcal{A}_U$ with two configurations κ_1, κ_2 such that U accepts a word iff $\kappa_1 \not\sim \kappa_2$. $\mathcal{A}_U$ is constructed to induce a bisimulation game in which Attacker gets a chance to choose a word to be accepted by U and simulate an accepting run (if one exists). It consists of two nearly identical components, which are linked by the Defender Forcing circuit in places. Other differences between them stem from the need to arrange for non-bisimilarity, in cases when the bisimulation game reaches a stage indicating acceptance or Attacker tried to cheat while simulating a run. We sketch the design of the components.

Input stage. Initially, we want Attacker to start choosing input letters and pushing them on the stack. This is to continue until Attacker decides to finish the input phase. Defender will simply copy the moves in other component. Technically, both kinds of choices can be implemented by deterministic push transitions that cover the range of input in both components. Observe that, in order to win (uncover non-bisimilarity), Attacker will eventually need to abandon the input stage to avoid infinite copying.

Transitions. Once the input phase is over, the automaton enters the simulation stage. Recall that the input word chosen by Attacker will be available on the stack in both components. The top of the stack will play the role of the head of U and we can use the two registers of $\mathcal{A}_U$ to emulate the two registers of U. To make transitions, we need to be able to access the tag at the top of the stack as well as compare the corresponding data value with the content of registers. The only way of inspecting

the top of the stack is by popping, but then we could lose the data value if it does not already occur in a register (the value might be needed later, e.g. the automaton might want to move it into a register). To avoid such a loss, we will let Attacker guess the outcome of the comparisons. However, Defender will be allowed to verify the correctness of such guesses (via Defender Forcing). During the verification the top of the stack will indeed be popped, but we shall be no longer concerned about losing it, because it will survive in a different branch of the game, which will carry on simulating the run. In order to implement the detection of incorrect guesses, we will need to break symmetry between the components and arrange for non-bisimilarity if Attacker's guess is correct.

Universal states. To simulate these, we can delegate the choice to Defender through Forcing. This will allow Defender to direct the game towards a failing branch, if one exists.

Head movements. To advance the tape, we simply use one of the pop-instructions.

Register reassignment. To move the currently scanned data value into a register, let us assume that the symbol is not in a register yet. Then we can refresh the content of the relevant register (to guess the data value at the top of the stack) and then perform a pop. Note that a wrong guess by Attacker will lead to a deadlock (no ability to pop), which gives Attacker the necessary incentive to guess correctly.

Accepting/rejecting states. If the simulation reaches a rejecting state, we arrange for bisimilarity (to attract Defender there). In accepting states, we arrange non-bisimilarity.

B. Detailed argument

Given a URA$_2$ $U = \langle \Sigma, Q, q_I, 2, \delta \rangle$, we shall construct a 2-VRPDA $\mathcal{A}_U$ such that $\kappa_1 \sim \kappa_2$ if and only if U does not accept any input, where $\kappa_j = (init^j, \tau_I, \epsilon)$ $(j = 1, 2)$ and $init^1, init^2$ are states. $\mathcal{A}_U$ will be constructed so as to induce a bisimulation game in which Attacker gets a chance to choose a word to be accepted and simulate an accepting run (if one exists). Without loss of generality, we shall assume injectivity of register assignments and that, whenever $\downarrow_r$ is used, the $\mathcal{D}$-value on the tape is not present in registers (these conditions can be enforced by modifying the transition function with the help of the finite control and appropriate book-keeping). Moreover, to avoid complications with borderline cases, we shall assume that U does not accept the empty word.

$\mathcal{A}_U$ will consist of two mostly identical components involving superscripted states from U as well as a number of auxiliary states implicit in the definitions below. The only connections between the two components will be due to the use of the Defender Forcing circuit. The only differences between the components will stem from the need to arrange for non-bisimilarity, in cases when the bisimulation game reaches a stage indicating acceptance or when Attacker makes a simulation mistake.

Below we explain the design of $\mathcal{A}_U$ at various stages of simulating U. We use arrows to define transitions according to the following conventions.

- $q_1 \xrightarrow{(t,l)/(t',j)} q_2$ stands for $(q_1, t, l, t', j, q_2) \in \delta_C$
- $q_1 \xrightarrow{(t,l)} q_2$ stands for $(q_1, t, l, q_2) \in \delta_N$
- $q_1 \xrightarrow{(t,l),(t',j)} q_2$ stands for $(q_1, t, l, t', j, q_2) \in \delta_R$

Given $q \in Q$, we write q^j $(j = 1, 2)$ for its superscripted variants to be included in $\mathcal{A}_U$. We shall rely on the following sets of tags.

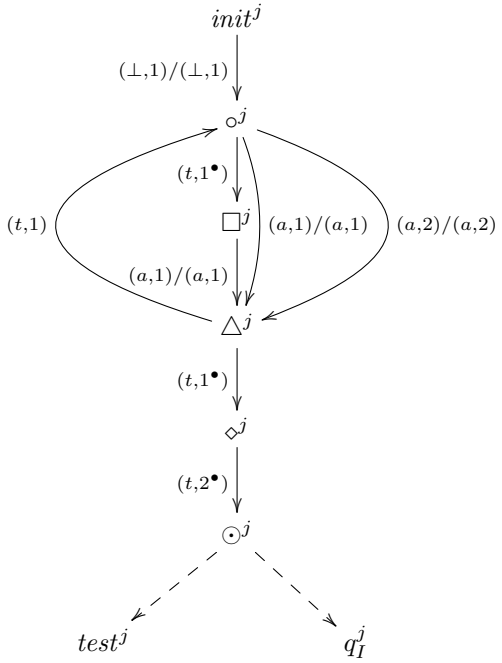
$$\begin{array}{rcl} \Sigma_C & = & \{\bot\} + \Sigma \\ \Sigma_N & = & \{t, t_1, t_2\} \\ \Sigma_R & = & \{t_R\} \end{array}$$

For the stack alphabet, we shall have $\Gamma = \Sigma_C$.

We start off by introducing new states $init^1, init^2$ that will be used to start the initial phase in which Attacker can choose an input word and push it on the stack.

Input Phase

When drawing a diagram featuring states superscripted with j, we mean to say that *two* copies of the design should be included into $\mathcal{A}_\mathcal{U}$, one for $j = 1$ and another for $j = 2$. We use $\circ, \Box, \triangle, \diamond, \odot$ to indicate auxiliary states to be included in each component. We shall reuse them in different cases on the understanding that they refer to *different* states in each case.



a ranges over Σ above. Consequently, if the bisimulation game starts from (κ_1, κ_2) then the above design gives Attacker a chance to pick a data word and push it on the stack. The three outgoing transitions from state $\circ^j$ correspond to (from left to right) Attacker picking for the next data value: a fresh data value not currently in either register, the data value currently stored in register 1 or the data value currently stored in register 2. The stack content in both copies will be the same. Attacker also decides when to end the input selection phase and proceed to $(\diamond^1, \diamond^2)$. The transition sequence $(t, 1^\bullet)(t, 2^\bullet)$ is intended

to give Attacker a chance to pick the right initial register assignment to support the simulation. For a match with URA, we need the initial values to be different from any data values present in the selected input word. Once Attacker generates the values and $(\odot^1, \odot^2)$ is reached, Defender will have an option to challenge the choice or to proceed with the simulation to (q_I^1, q_I^2). This will be achieved through Defender Forcing, represented by dashed lines. We shall return to the exact design of $test^j$, after we apply Defender Forcing in simpler cases.

The subsequent part of the construction corresponds to checking that the selected word is accepted (we want Attacker to win iff this is the case). We analyze each kind of transition in turn.

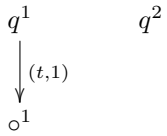
Transitions

1) $\delta(q) = \bot$ *(rejection):*

$$q^j$$

We do not add any transitions from q^1 or q^2. This ensures bisimilarity, should the game enter configurations with states q^1, q^2 respectively.

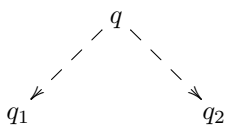
2) $\delta(q) = \top$ *(acceptance):*



Note that we do not add any transitions from q^2 in order to generate non-bisimilar configurations.

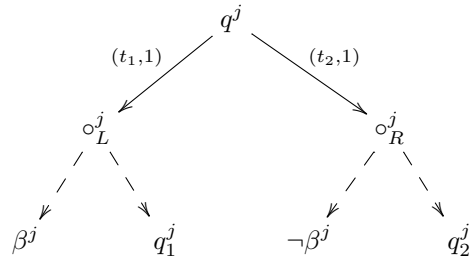
3) $\delta(q) = q_1 \wedge q_2$ *(universal choice):* We will let Defender choose the state (q_1 or q_2) that should be pursued. Note that this is consistent with the goal of relating emptiness with bisimilarity. To that end, we use the Defender Forcing circuit from Section II-A, which we write as $DF(\kappa^1, \kappa^2, \kappa_1^1, \kappa_1^2, \kappa_2^1, \kappa_2^2)$. In our particular case, we need to introduce transitions to generate the graph $DF(q^1, q^2, q_1^1, q_1^2, q_2^1, q_2^2)$. Recall that in order for the technique to work with VPDRA, we need to be sure that the stacks and registers are used in the same way by each of the components. This is an easily verifiable property of our constructions. In order to implement DF we need two different labels, e.g. $(t_1, 1)$ and $(t_2, 1)$.

For brevity, in what follows, we shall write



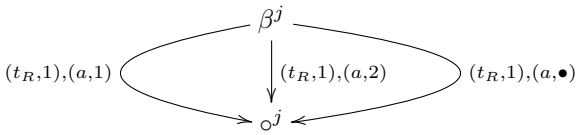
to refer to the use of $DF(q^1, q^2, q_1^1, q_1^2, q_2^1, q_2^2)$.

4) $\delta(q) = q_1 \triangleleft \beta \triangleright q_2$*:* Here we shall let Attacker choose between q_1 and q_2 but the Defender will later be able to challenge the decision (and check whether it is consistent with β). For this purpose we use

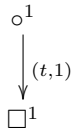


where $\beta^1, \beta^2, \neg\beta^1, \neg\beta^2$ will be constructed so that the first two induce bisimilarity iff β fails and the last two induce bisimilarity iff β holds. We do case analysis on β.

$\beta = a$ *(stack tag comparison).* To handle β^j, we introduce

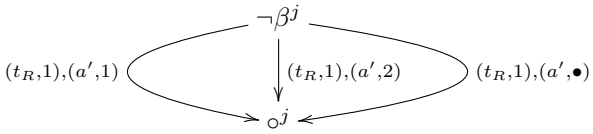


and

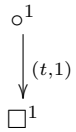


We explain the idea behind this first gadget, the rest are similar. If Defender was correct to challenge Attacker because Attacker cheated, i.e. the letter under the head (top of stack) is not tagged by a (despite Attacker's claim), then Attacker will not be able to play any transition from β_j and hence Defender will win. If Defender challenged Attacker incorrectly, then Attacker will be able to play exactly one of the transitions, according to the current register assignment, and Defender will copy the move. However, in the following move Attacker will win, since Attacker will play the only transition out of $\circ^1$ and Defender cannot match this in $\circ^2$, since it has no available transitions.

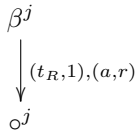
For $\neg\beta^j$ we can take



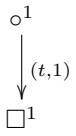
where a' ranges over $\Sigma \setminus \{a\}$, and:



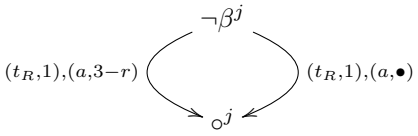
$\beta = \uparrow_r$ *(stack $\mathcal{D}$-value comparison).* To handle β^j, we introduce



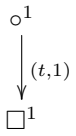
with a ranging over Σ, and:



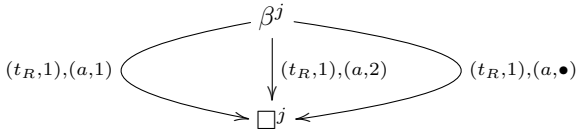
For $\neg\beta^j$ we can take



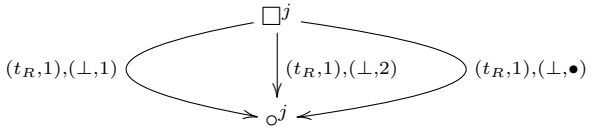
with a ranging over Σ, and



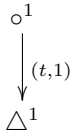
$\beta = \mathsf{end}$ *(last tape-symbol)*. To handle β^j, we introduce



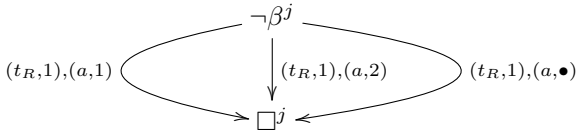
with a ranging over Σ,



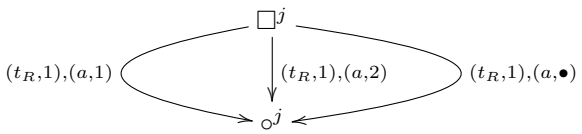
and:



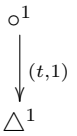
For $\neg\beta^j$ we can take



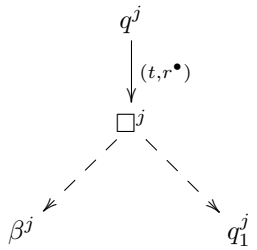
with a ranging over Σ,



with a ranging over Σ again, and:

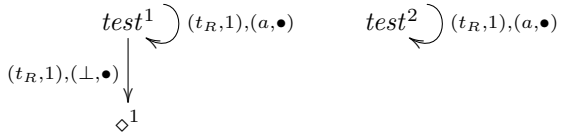


5) $\delta(q) = \downarrow_r q_1$*:* We add

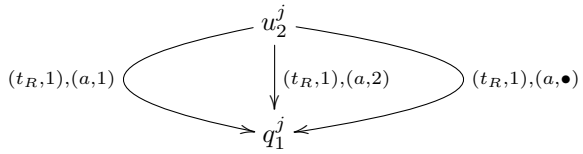


and also add outgoing transitions for β^j, as for the case $\beta = \uparrow_r$. Note that the arrangement forces Attacker to guess the $\mathcal{D}$-value stored on top of the stack (and place it in register r).

6) Freshness testing ($test^j$*):* We design $test^1$ and $test^2$ in such a way that they will lead to non-bisimilarity iff Attacker guessed an initial register assignment that does not contain any data values encountered during the input phase. a ranges over Σ.



7) $\delta(q) = Xq_1$ *(move head right/reject):* To take advantage of previous cases, we represent the transition as $u_1 \triangleleft \mathsf{end} \triangleright u_2$ with $\delta(u_1) = \bot$ and $\delta(u_2) = Xq_1$. This makes sure that X is only invoked when we are not at the end of the word. Consequently, we can reuse the previous constructions for $u_1 \triangleleft \beta \triangleright u_2$ and $\bot$ cases. To handle u_2, we can now add



with a ranging over Σ.

8) $\delta(q) = \overline{X}q_1$ *(move head right/accept):* This is nearly the same as the previous case: now we decompose the transition into $u_1 \triangleleft \mathsf{end} \triangleright u_2$ with $\delta(u_1) = \top$ and $\delta(u_2) = Xq_1$.

Lemma 72. *$\kappa_1 \sim \kappa_2$ if and only if U does not accept any words.*

This implies Theorem 46.