Bridging the Gap between Fair Simulation and Trace Inclusion*

Yonit Kesten[1], Nir Piterman[2], and Amir Pnueli[2]

[1] Ben Gurion University, Beer-Sheva, Israel. ykesten@bgumail.bgu.ac.il
[2] Weizmann Institute, Rehovot, Israel. (nirp,amir)@wisdom.weizmann.ac.il

Abstract. The paper considers the problem of checking abstraction between two finite-state *fair discrete systems*. In automata-theoretic terms this is trace inclusion between two Streett automata. We propose to reduce this problem to an algorithm for checking fair simulation between two generalized Büchi automata. For solving this question we present a new triply nested μ-calculus formula which can be implemented by symbolic methods.
We then show that every trace inclusion of this type can be solved by fair simulation, provided we augment the concrete system (the contained automaton) by appropriate auxiliary variables. This establishes that fair simulation offers a complete method for checking trace inclusion. We illustrate the feasibility of the approach by algorithmically checking abstraction between finite state systems whose abstraction could only be verified by deductive methods up to now.

1 Introduction

A frequently occurring problem in verification of reactive systems is the problem of *abstraction* (symmetrically *refinement*) in which we are given a concrete reactive system C and an abstract reactive system A and are asked to check whether A *abstracts* C, denoted $C \sqsubseteq A$. In the linear-semantics framework this question calls for checking whether any observation of C is also an observation of A. For the case that both C and A are finite-state systems with weak and strong fairness this problem can be reduced to the problem of language inclusion between two Streett automata (e.g., [Var91]).

In theory, this problem has an exponential-time algorithmic solution based on the complementation of the automaton representing the abstract system A [Saf92]. However, the complexity of this algorithm makes its application prohibitively expensive. For example, our own interest in the finite-state abstraction problem stems from applications of the verification method of *network invariants* [KP00a,KPSZ02,WL89]. In a typical application of this method, we are asked to verify the abstraction $P_1 \parallel P_2 \parallel P_3 \parallel P_4 \sqsubseteq P_5 \parallel P_6 \parallel P_7$, claiming that 3 parallel copies of the dining philosophers process abstract 4 parallel copies of the same process. The system on the right has about 1800 states. Obviously, to complement a Streett automaton of 1800 states is hopelessly expensive.

A partial but more effective solution to the problem of checking abstraction between systems (trace inclusion between automata) is provided by the notion of *simulation*.

* This research was supported in part by THE ISRAEL SCIENCE FOUNDATION (grant no.106/02-1) and the John von-Neumann Minerva center for Verification of Reactive Systems.

W.A. Hunt, Jr. and F. Somenzi (Eds.): CAV 2003, LNCS 2725, pp. 381–393, 2003.

Introduced first by Milner [Mil71], we say that system A simulates system C, denoted $C \preceq A$, if there exists a *simulation relation* R between the states of C and the states of A. It is required that if $(s_C, s_A) \in R$ and system C can move from state s_C to state s'_C, then system A can move from s_A to some s'_A such that $(s'_C, s'_A) \in R$. Additional requirements on R are that if $(s_C, s_A) \in R$ then s_C and s_A agree on the values of their observables, and for every s_C initial in C there exists s_A initial in A such that $(s_C, s_A) \in R$. It is obvious that $C \preceq A$ is a sufficient condition for $C \sqsubseteq A$. For finite-state systems, we can check $C \preceq A$ in time proportional to $(|\Sigma_C| \cdot |\Sigma_A|)^2$ where Σ_C and Σ_A are the sets of states of A and C respectively [BR96,HHK95].

While being a sufficient condition, simulation is definitely not a necessary condition for abstraction. This is illustrated by the two systems presented in Fig. 1

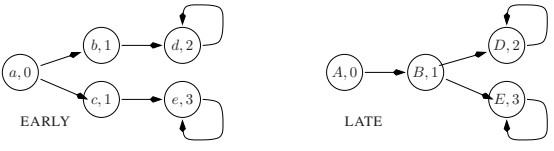

Fig. 1. Systems EARLY and LATE

The labels in these two systems consist of a local state name (a–e, A–E) and an observable value. Clearly these two systems are (observation)-equivalent because they each have the two possible observations $012^\omega + 013^\omega$. Thus, each of them abstracts the other. However, when we examine their simulation relation, we find that EARLY $\preceq$ LATE but LATE $\not\preceq$ EARLY. This example illustrates that, in some cases we can use simulation in order to establish abstraction (trace inclusion) but this method is not complete.

The above discussion only covered the case that C and A did not have any fairness constraints. There were many suggestions about how to enhance the notion of simulation in order to account for fairness [GL94,LT87,HKR97,HR00]. The one we found most useful for our purposes is the definition of fair simulation from [HKR97]. Henzinger et al. proposed a game-based view of simulation. As in the unfair case, the definition assumes an underlying simulation relation R which implies equality of the observables. However, in the presence of fairness, it is not sufficient to guarantee that every step of the concrete system can be matched by an abstract step with corresponding observables. Here we require that the abstract system has a *strategy* such that any joint run of the two systems, where the abstract player follows this strategy either satisfies the fairness requirements of the abstract system or fails to satisfy the fairness requirements of the concrete system. This guarantees that every concrete observation has a corresponding abstract observation with matching values of the observables.

Algorithmic Considerations. In order to determine whether one system fairly simulates another (*solve fair simulation*) we have to solve games [HKR97]. When the two systems in question are reactive systems with strong fairness (Streett), the winning condition of the resulting game is an implication between two Streett conditions (*fsim-games*). In [HKR97] the solution of fsim-games is reduced to the solution of Streett games. In [KV98] an algorithm for solving Streett games is presented. The time complexity of this

approach is $(|\Sigma_A| \cdot |\Sigma_C| \cdot (3^{k_A} + k_C))^{2k_A+k_C} \cdot (2k_A + k_C)!$ where k_C and k_A denote the number of Streett pairs of C and A. Clearly, this complexity is too high. It is also not clear whether this algorithm can be implemented symbolically.

In the context of fair simulation, Streett systems cannot be reduced to simpler systems [KPV00]. That is, in order to solve the question of fair simulation between Streett systems we have to solve fsim-games in their full generality. However, we are only interested in fair simulation as a precondition for trace inclusion. In the context of trace inclusion we can reduce the problem of two reactive systems with strong fairness to an equivalent problem with weak fairness. Formally, for the reactive systems C and A with Streett fairness requirements, we construct C^B and A^B with generalized Büchi requirements, such that $C \sqsubseteq A$ iff $C^B \sqsubseteq A^B$. Solving fair simulation between C^B and A^B is simpler. The winning condition of the resulting game is an implication between two generalized Büchi conditions (denoted *generalized Streett[1]*).

In [dAHM01], a solution for games with winning condition expressed as a general LTL formula is presented. The algorithm in [dAHM01] constructs a deterministic parity word automaton for the winning condition. The automaton is then converted into a μ-calculus formula that evaluates the set of winning states for the relevant player.

In [EL86], Emerson and Lei show that a μ-calculus formula is in fact a recipe for symbolic model checking [1]. The main factor in the complexity of μ-calculus model checking is the *alternation depth* of the formula. The symbolic algorithm for model checking a μ-calculus formula of alternation depth k takes time proportional to $(mn)^k$ where m is the size of the formula and n is the size of the model [EL86].

In fsim-games the winning condition is an implication between two Streett conditions. A deterministic Streett automaton for this winning condition has $3^{k_A} \cdot k_C$ states and $2k_A + k_C$ pairs. A deterministic parity automaton for the same condition has $3^{k_A} \cdot k_C \cdot (2k_A + k_C)!$ states and index $4k_A + 2k_C$. The μ-calculus formula constructed by [dAHM01] is of alternation depth $4k_A + 2k_C$ and proportional in size to $3^{k_A} \cdot k_C \cdot (2k_A + k_C)!$. Hence, in this case, there is no advantage in using [dAHM01].

In the case of generalized Streett[1] games, a deterministic parity automaton for the winning condition has $|J_C| \cdot |J_A|$ states and index 3, where $|J_C|$ and $|J_A|$ denote the number of Büchi sets in the fairness of C^B and A^B respectively. The μ-calculus formula of [dAHM01] is proportional to $3|J_C| \cdot |J_A|$ and has alternation depth 3.

We give an alternative μ-calculus formula that solves generalized Streett[1] games. Our formula is also of alternation depth 3 but its length is proportional to $2|J_C| \cdot |J_A|$ and it is simpler than that of [dAHM01]. Obviously, our algorithm is tailored for the case of generalized-Streett[1] games while [dAHM01] give a generic solution for any LTL game [2]. The time complexity of solving fair simulation between two reactive systems after converting them to systems with generalized Büchi fairness requirements is $(|\Sigma_A| \cdot |\Sigma_C| \cdot 2^{k_A+k_C} \cdot (|J_A| + |J_C| + k_A + k_C))^3$.

[1] There are more efficient algorithms for μ-calculus model checking [Jur00]. However, Jurdzinski's algorithm cannot be implemented symbolically.

[2] One may ask why not take one step further and convert the original reactive systems to Büchi systems. In this case, the induced game is a parity[3] game and there is a simple algorithm for solving it. Although both algorithms work in cubic time, the latter performed much worse than the one described above.

Making the Method Complete. Even if we succeed to present a complexity-acceptable algorithm for checking fair simulation between generalized-Büchi systems, there is still a major drawback to this approach which is its incompleteness. As shown by the example of Fig. 1, there are (trivially simple) systems C and A such that $C \sqsubseteq A$ but this abstraction cannot be proven using fair simulation. Fortunately, we are not the first to be concerned by the incompleteness of simulation as a method for proving abstraction. In the context of infinite-state system verification, Abadi and Lamport studied the method of simulation using an *abstraction mapping* [AL91]. It is not difficult to see that this notion of simulation is the infinite-state counterpart of the fair simulation as defined in [HKR97] but restricted to the use of memory-less strategies. However, [AL91] did not stop there but proceeded to show that if we are allowed to add to the concrete system auxiliary *history* and *prophecy* variables, then the simulation method becomes *complete*. That is, with appropriate augmentation by auxiliary variables, every abstraction relation can be proven using fair simulation. History variables remove the restriction to memory-less strategies, while prophecy variables allow to predict the future and use fair simulation to establish, for example, the abstraction LATE $\sqsubseteq$ EARLY.

The application of Abadi-Lamport, being deductive in nature, requires the users to decide on the appropriate history and prophecy variables, and then design their abstraction mapping which makes use of these auxiliary variables. Implementing these ideas in the finite-state (and therefore algorithmic) world, we expect the strategy (corresponding to the abstraction mapping) to be computed fully automatically. Thus, in our implementation, the user is still expected to identify the necessary auxiliary history or prophecy variables, but following that, the rest of the process is automatic. For example, wishing to apply our algorithm in order to check the abstraction LATE $\sqsubseteq$ EARLY, the user has to specify the augmentation of the concrete system by a temporal tester for the LTL formula $\diamondsuit(x = 2)$. Using this augmentation, the algorithm manages to prove that the augmented system (LATE +tester) is fairly simulated (hence abstracted) by EARLY.

In summary, the contributions of this paper are:

1. Suggesting the usage of fair simulation as a precondition for abstraction between two reactive systems (Streett automata).
2. Observing that in the context of fair simulation for checking abstraction we can simplify the game acceptance condition from implication between two Streett conditions to implication between two generalized Büchi conditions.
3. Providing a more efficient μ-calculus formula and its implementation by symbolic model-checking tools for solving the fair simulation between two generalized Büchi systems.
4. Claiming and demonstrating the completeness of the fair-simulation method for proving abstraction between two systems, at the price of augmenting the concrete system by appropriately chosen "observers" and "testers".

2 The Computational Model

As a computational model, we take the model of *fair discrete system* (FDS) [KP00b]. An FDS $\mathcal{D} : \langle V, \mathcal{O}, \Theta, \rho, \mathcal{J}, \mathcal{C} \rangle$ consists of the following components.

- $V = \{u_1, ..., u_n\}$: A finite set of typed *state variables* over finite domains. A *state* s is a type-consistent interpretation of V. We denote by Σ the set of all states.
- $\mathcal{O} \subseteq V$: A subset of externally *observable variables*.
- Θ : The *initial condition*. An assertion characterizing all the initial states.
- ρ : A *transition relation*. This is an assertion $\rho(V, V')$, relating a state $s \in \Sigma$ to its $\mathcal{D}$-successor $s' \in \Sigma$ by referring to both unprimed and primed versions of the state variables. State s' is a $\mathcal{D}$-*successor* of s if $\langle s, s' \rangle \models \rho(V, V')$, where $\langle s, s' \rangle$ is the joint interpretation which interprets $x \in V$ as $s[x]$, and x' as $s'[x]$.
- $\mathcal{J} = \{J_1, \ldots, J_k\}$: Assertions expressing the *justice* (*weak fairness*) requirements.
- $\mathcal{C} = \{\langle p_1, q_1 \rangle, \ldots \langle p_n, q_n \rangle\}$: Assertions expressing the *compassion* (*strong fairness*) requirements .

We require that every state $s \in \Sigma$ has at least one $\mathcal{D}$-successor. This is ensured by including in ρ the *idling* disjunct $V' = V$. Let $\sigma : s_0, s_1, ...,$ be a sequence of states, φ be an assertion, and $j \geq 0$ be a natural number. We say that j is a φ-position of σ if s_j is a φ-state. Let $\mathcal{D}$ be an FDS for which the above components have been identified. A *run* of $\mathcal{D}$ is an infinite sequence of states $\sigma : s_0, s_1, ...,$ satisfying following:

- *Initiality:* s_0 is initial, i.e., $s_0 \models \Theta$.
- *Consecution:* For $j = 0, 1, ...,$ the state s_{j+1} is a $\mathcal{D}$-successor of the state s_j.

A run of $\mathcal{D}$ is called a *computation* if it satisfies the following:

- *Justice:* For each $J \in \mathcal{J}$, σ contains infinitely many J-positions
- *Compassion:* For each $\langle p, q \rangle \in \mathcal{C}$, if σ contains infinitely many p-positions, it must also contain infinitely many q-positions.

Let *runs*$(\mathcal{D})$ denote the set of runs of $\mathcal{D}$ and $\mathcal{C}omp(\mathcal{D})$ the set of computations of $\mathcal{D}$.

Systems $\mathcal{D}_1$ and $\mathcal{D}_2$ are *compatible* if the intersection of their variables is observable in both systems. For compatible systems $\mathcal{D}_1$ and $\mathcal{D}_2$, we define their *asynchronous parallel composition*, denoted by $\mathcal{D}_1 \parallel \mathcal{D}_2$, and the *synchronous parallel composition*, denoted by $\mathcal{D}_1 \,|||\, \mathcal{D}_2$, in the usual way [KP00a]. The primary use of synchronous composition is for combining a system with a *tester* T_φ for an LTL formula φ.

The *observations* of $\mathcal{D}$ are the projection $\mathcal{D} \Downarrow_{\mathcal{O}}$ of $\mathcal{D}$-computations onto $\mathcal{O}$. We denote by *Obs*$(\mathcal{D})$ the set of all observations of $\mathcal{D}$. Systems $\mathcal{D}_C$ and $\mathcal{D}_A$ are said to be *comparable* if there is a one to one correspondence between their observable variables. System $\mathcal{D}_A$ is said to be an *abstraction* of the comparable system $\mathcal{D}_C$, denoted $\mathcal{D}_C \sqsubseteq \mathcal{D}_A$, if *Obs*$(\mathcal{D}_C) \subseteq$ *Obs*$(\mathcal{D}_A)$. The abstraction relation is reflexive and transitive. It is also *property restricting*. That is, if $\mathcal{D}_C \sqsubseteq \mathcal{D}_A$ then $\mathcal{D}_A \models p$ implies that $\mathcal{D}_C \models p$ for an LTL property p. We say that two comparable FDS's $\mathcal{D}_1$ and $\mathcal{D}_2$ are *equivalent*, denoted $\mathcal{D}_1 \sim \mathcal{D}_2$ if *Obs*$(\mathcal{D}_1) =$ *Obs*$(\mathcal{D}_2)$. For compatibility with automata terminology, we refer to the observations of $\mathcal{D}$ also as the *traces* of $\mathcal{D}$.

All our concrete examples are given in SPL, which is used to represent concurrent programs (e.g., [MP95,MAB$^+$94]). Every SPL program can be compiled into an FDS in a straightforward manner.

From FDS **to** JDS**.** An FDS with no compassion requirements is called a *just discrete system* (JDS). Let $\mathcal{D} : \langle V, \mathcal{O}, \Theta, \rho, \mathcal{J}, \mathcal{C} \rangle$ be an FDS such that $\mathcal{C} = \{(p_1, q_1), \ldots, (p_m, q_m)\}$ and $m > 0$. We define a JDS $\mathcal{D}^B : \langle V^B, \mathcal{O}^B, \Theta^B, \rho^B, \mathcal{J}^B, \emptyset \rangle$ equivalent to $\mathcal{D}$, as follows:

- $V^{B} = V \cup \{n\_p_i : \textbf{Boolean} \mid (p_i, q_i) \in \mathcal{C}\} \cup \{x_c\}$.
- $\mathcal{O}^{B} = \mathcal{O}$.
- $\Theta^{B} = \Theta \ \wedge \ x_c = 0 \ \wedge \ \bigwedge_{(p_i,q_i)\in\mathcal{C}} n\_p_i = 0$.
- $\rho^{B} = \rho \ \wedge \ \rho_{n\_p} \ \wedge \ \rho_c$, where

$$\rho_{n\_p} : \bigwedge_{(p_i,q_i)\in\mathcal{C}} (n\_p_i \rightarrow n\_p'_i) \qquad \rho_c : x'_c = (x_c \ \vee \ \bigvee_{(p_i,q_i)\in\mathcal{C}} (p_i \ \wedge \ n\_p_i))$$

- $\mathcal{J}^{B} = \mathcal{J} \cup \{\neg x_c\} \cup \{n\_p_i \ \vee \ q_i \mid (p_i, q_i) \in \mathcal{C}\}$.

The transformation of an FDS to a JDS follows the transformation of Streett automata to generalized Büchi Automata (see [Cho74] for finite state automata and [Var91] for infinite state automata). We add one Boolean variable n_p_i per compassion requirement. This variable is initialized to 0, it can be set nondeterministically to 1 and is never reset. The nondeterministic setting is in fact a guess that no more p_i states are encountered. Accordingly, we add the justice $n\_p_i \vee q_i$ so either n_p_i is set (and p_i is false from that point) or q_i is visited infinitely often. We add one additional variable x_c initialized to 0, set to 1 at a point satisfying $\bigvee_{i=1}^{m}(p_i \wedge n\_p_i)$ and never reset. Once x_c is set it indicates a *mis-prediction*. We guessed wrong that some p_i never holds anymore. We add the justice requirement $\neg x_c$ to ensure that a run in which x_c is set, is not a computation.

3 Simulation Games

Let $\mathcal{D}_C : \langle V_C, \mathcal{O}_C, \Theta_C, \rho_C, \mathcal{J}_C, \mathcal{C}_C \rangle$ and $\mathcal{D}_A : \langle V_A, \mathcal{O}_A, \Theta_A, \rho_A, \mathcal{J}_A, \mathcal{C}_A \rangle$ be two comparable FDS's. We denote by Σ_C and Σ_A the sets of states of $\mathcal{D}_C$ and $\mathcal{D}_A$ respectively. We define the *simulation game structure* (SGS) associated with $\mathcal{D}_C$ and $\mathcal{D}_A$ to be the tuple $G : \langle \mathcal{D}_C, \mathcal{D}_A \rangle$. A *state* of G is a type-consistent interpretation of the variables in $V_C \cup V_A$. We denote by Σ_G the set of states of G. We say that a state $s \in \Sigma_G$ is *correlated*, if $s[\mathcal{O}_C] = s[\mathcal{O}_A]$. We denote by $\Sigma_{cor} \subset \Sigma_G$ the subset of correlated states.

For two states s and t of G, t is an A-successor of s if $(s, t) \models \rho_A$ and $s[V_C] = t[V_C]$. Similarly, t is a C-successor of s if $(s, t) \models \rho_C$ and $s[V_A] = t[V_A]$. A *run* of G is a maximal sequence of states $\sigma : s_0, s_1, \ldots$ satisfying the following:

- *Consecution:* For each $j = 0, ...,$ ■ s_{2j+1} is a C-successor of s_{2j}.
  ■ s_{2j+2} is a A-successor of s_{2j+1}.
- *Correlation:* For each $j = 0, ...,$ ■ $s_{2j} \in \Sigma_{cor}$

We say that a run is *initialized* if it satisfies

- *Initiality:* $s_0 \models \Theta_A \ \wedge \ \Theta_C$

Let G be an SGS and σ be a run of G. The run σ can be viewed as a two player game. Player C, represented by $\mathcal{D}_C$, taking ρ_C transitions from even numbered states and player A, represented by $\mathcal{D}_A$, taking ρ_A transitions from odd numbered states. The observations of the two players are correlated on all even numbered states of a run.

A run σ is *winning for player* A if it is infinite and either $\sigma \Downarrow_{V_C}$ is not a computation of $\mathcal{D}_C$ or $\sigma \Downarrow_{V_A}$ is a computation of $\mathcal{D}_A$, i.e. if $\sigma \models \mathcal{F}_C \rightarrow \mathcal{F}_A$, where for $\eta \in \{A, C\}$,

$$\mathcal{F}_\eta : \bigwedge_{J\in\mathcal{J}_\eta} \Box \Diamond J \ \wedge \ \bigwedge_{(p,q)\in\mathcal{C}_\eta} (\Box \Diamond p \rightarrow \Box \Diamond q).$$

Otherwise, σ is *winning for player* C.

Let D_A and D_C be some finite domains, intended to record facts about the past history of a computation (serve as a memory). A *strategy* for player A is a partial function $f_A : D_A \times \Sigma_G \mapsto D_A \times \Sigma_{cor}$ such that if $f_A(d, s) = (d', s')$ then s' is an A-successor of s'. A *strategy* for player C is a partial function $f_C : D_C \times \Sigma_{cor} \mapsto D_C \times \Sigma_G$ such that if $f_C(d, s) = (d', s')$ then s' is a C-successor of s. Let f_A be a strategy for player A, and $s_0 \in \Sigma_{cor}$. A run $s_0, s_1, \ldots$ is said to be *compliant* with strategy f_A if there exists a sequence of D_A-values $d_0, d_2, \ldots, d_{2j}, \ldots$ such that $(d_{2j+2}, s_{2j+2}) = f_A(d_{2j}, s_{2j+1})$ for every $j \geq 0$. Strategy f_A is *winning* for player A from state $s \in \Sigma_{cor}$ if all s-runs (runs departing from s) which are compliant with f_A are winning for A. A winning strategy for player C is defined similarly. We denote by W_A the set of states from which there exists a winning strategy for player A. The set W_C is defined similarly.

An SGS G is called *determinate* if the sets W_A and W_C define a partition on Σ_{cor}. It is well known that every SGS is determinate [GH82].

3.1 μ-Calculus

We define μ-calculus [Koz83] over game structures. Consider two FDS's $\mathcal{D}_C : \langle V_C, \mathcal{O}_C, \Theta_C, \rho_C, \mathcal{J}_C, \mathcal{C}_C \rangle$, $\mathcal{D}_A : \langle V_A, \mathcal{O}_A, \Theta_A, \rho_A, \mathcal{J}_A, \mathcal{C}_A \rangle$ and the SGS $G : \langle \mathcal{D}_C, \mathcal{D}_A \rangle$. For every variable $v \in V_C \cup V_A$ the formula $v = i$ where i is a constant that is type consistent with v is an *atomic formula* (p). Let $V = \{X, Y, \ldots\}$ be a set of *relational variables*. Each relational variable can be assigned a subset of Σ_{cor}. The μ-calculus formulas are constructed as follows.

$$f ::= p \mid \neg p \mid X \mid f \vee f \mid f \wedge f \mid \diamondsuit f \mid \boxdot f \mid \mu X f \mid \nu X f$$

A formula f is interpreted as the set of states in Σ_{cor} in which f is true. We write such set of states as $[[f]]^e_G$ where G is the SGS and $e : V \rightarrow 2^{\Sigma_{cor}}$ is an *environment*. The set $[[f]]^e_G$ is defined for the operators $\diamondsuit$ and $\boxdot$ as follows.

- $[[\diamondsuit f]]^e_G = \{s \in \Sigma_{cor} \mid \forall t,\ (s,t) \models \rho_C \rightarrow \exists s',\ (t,s') \models \rho_A \text{ and } s' \in [[f]]^e_G\}$.
- $[[\boxdot f]]^e_G = \{s \in \Sigma_{cor} \mid \exists t,\ (s,t) \models \rho_C \text{ and } \forall s',\ (t,s') \models \rho_A \rightarrow s' \in [[f]]^e_G\}$.

For the rest of the operators the semantics is as in the usual definition [Eme97]. The *alternation depth* of a formula is the number of alternations in the nesting of least and greatest fixpoints. A μ-calculus formula defines a symbolic algorithm for computing $[[f]]$ [EL86]. For a μ-calculus formula of alternation depth k, the run time of this algorithm is $|\Sigma_{cor}|^k$. For a full exposition of μ-calculus we refer the reader to [Eme97].

4 Trace Inclusion and Fair Simulation

In the following, we summarize our solution to verifying abstraction between two FDS's systems, or equivalently, trace inclusion between two Streett automata.

Let $\mathcal{D}_C : \langle V_C, \mathcal{O}_C, \Theta_C, \rho_C, \mathcal{J}_C, \mathcal{C}_C \rangle$ and $\mathcal{D}_A : \langle V_A, \mathcal{O}_A, \Theta_A, \rho_A, \mathcal{J}_A, \mathcal{C}_A \rangle$ be two comparable FDS's. We want to verify that $\mathcal{D}_C \sqsubseteq \mathcal{D}_A$. The best algorithm for solving abstraction is exponential [Saf92]. We therefore advocate to verify fair simulation

[HKR97] as a precondition for abstraction. We adopt the definition of fair simulation presented in [HKR97]. Given $\mathcal{D}_C$ and $\mathcal{D}_A$, we form the SGS $G : \langle \mathcal{D}_C, \mathcal{D}_A \rangle$. We say that $S \subseteq \Sigma_{cor}$ is a *fair-simulation* between $\mathcal{D}_A$ and $\mathcal{D}_C$ if there exists a strategy f_A such that every f_A-compliant run σ from a state $s \in S$ is winning for player A and every even state in σ is in S. We say that $\mathcal{D}_A$ *fairly-simulates* $\mathcal{D}_C$, denoted $\mathcal{D}_C \preceq_f \mathcal{D}_A$, if there exists a fair-simulation S such that for every state $s_C \in \Sigma_C$ satisfying $s_C \models \Theta_C$ there exists a state $t \in S$ such that $t \Downarrow_{V_C} = s_C$ and $t \models \Theta_A$.

Claim. [HKR97] If $\mathcal{D}_C \preceq_f \mathcal{D}_A$ then $\mathcal{D}_C \sqsubseteq \mathcal{D}_A$. The reverse implication does not hold.

It is shown in [HKR97] that we can determine whether $\mathcal{D}_C \preceq_f \mathcal{D}_A$ by computing the set $W_A \subseteq \Sigma_{cor}$ of states which are winning for A in the SGS G. If for every state $s_C \in \Sigma_c$ satisfying $s_C \models \Theta_C$ there exists some state $t \in W_A$ such that $t \Downarrow_{V_C} = s_C$ and $t \models \Theta_A$, then $\mathcal{D}_C \preceq_f \mathcal{D}_A$.

Let $k_C = |\mathcal{C}_C|$ (number of compassion requirements of $\mathcal{D}_C$), $k_A = |\mathcal{C}_A|$, $n = |\Sigma_C| \cdot |\Sigma_A| \cdot (3^{k_C} + k_A)$, and $f = 2k_C + k_A$.

Theorem 1. [HKR97,KV98] *We can solve fair simulation for $\mathcal{D}_C$ and $\mathcal{D}_A$ in time $O(n^{2f+1} \cdot f!)$.*

As we are interested in fair simulation as a precondition for trace inclusion, we take a more economic approach. Given two FDS's, we first convert the two to JDS's using the construction in Section 2. We then solve the simulation game for the two JDS's.

Consider the FDS's $\mathcal{D}_C$ and $\mathcal{D}_A$. Let $\mathcal{D}_C^B : \langle V_C^B, \mathcal{O}_C^B, \Theta_C^B, \rho_C^B, \mathcal{J}_C^B, \emptyset \rangle$ and $\mathcal{D}_A^B : \langle V_A^B, \mathcal{O}_A^B, \Theta_A^B, \rho_A^B, \mathcal{J}_A^B, \emptyset \rangle$ be the JDS's equivalent to $\mathcal{D}_C$ and $\mathcal{D}_A$. Consider the game $G : \langle \mathcal{D}_C^B, \mathcal{D}_A^B \rangle$. The winning condition for this game is: $\bigwedge_{J_C \in \mathcal{J}_C^B} J_C \rightarrow \bigwedge_{J_A \in \mathcal{J}_A^B} J_A$. We call such games *generalized Streett*[1] *games*. We claim that the formula in Equation (1) evaluates the set W_A of states winning for player A. Intuitively, the greatest fixpoint νX evaluates the set of states from which player A can control the run to remain in $\neg J_k^C$ states. The least fixpoint μY then evaluates the states from which player A in a finite number of steps controls the run to avoid one of the justice conditions J_k^C. This represents the set H of all states from which player A wins as a result of the run of $\mathcal{D}_C^B$ violating justice. Finally, the outermost greatest fixpoint νZ_j adds to H the states from which player A can force the run to satisfy the fairness requirement of $\mathcal{D}_A^B$.

$$\varphi = \nu \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_n \end{bmatrix} \begin{bmatrix} \mu Y \left(\vee_{k=1}^m \nu X (J_1^A \wedge \bigcirc Z_2 \quad \vee \quad \bigcirc Y \quad \vee \quad \neg J_k^C \wedge \bigcirc X) \right) \\ \mu Y \left(\vee_{k=1}^m \nu X (J_2^A \wedge \bigcirc Z_3 \quad \vee \quad \bigcirc Y \quad \vee \quad \neg J_k^C \wedge \bigcirc X) \right) \\ \vdots \\ \mu Y \left(\vee_{k=1}^m \nu X (J_n^A \wedge \bigcirc Z_1 \quad \vee \quad \bigcirc Y \quad \vee \quad \neg J_k^C \wedge \bigcirc X) \right) \end{bmatrix} \quad (1)$$

Claim. $W_A = [[\varphi]]$

The proof of the claim will appear in the full version.
Using the algorithm in [EL86] the set $[[\varphi]]$ can be evaluated symbolically.

Theorem 2. *The* SGS G *can be solved in time* $O((|\Sigma_C^B| \cdot |\Sigma_A^B| \cdot |\mathcal{J}_C^B| \cdot |\mathcal{J}_A^B|)^3)$.

To summarize, in order to use fair simulation as a precondition for trace inclusion we propose to convert the FDS's into JDS's and use the formula in Equation (1) to evaluate symbolically the winning set for player A.

Corollary 1. *Given* $\mathcal{D}_C$ *and* $\mathcal{D}_A$*, we can determine whether* $\mathcal{D}_C^B \preceq_f \mathcal{D}_A^B$ *in time proportional to* $O((|\Sigma_C| \cdot 2^{k_C} \cdot |\Sigma_A| \cdot 2^{k_A} \cdot (k_C + |\mathcal{J}_C| + k_A + |\mathcal{J}_A|))^3)$.

5 Closing the Gap

As discussed in the introduction, fair simulation implies trace inclusion but not the other way around. In [AL91], fair simulation is considered in the context of infinite-state systems. It is easy to see that the definition of *fair simulation* given in [AL91], is the infinite-state counterpart of fair simulation as defined in [HKR97], but restricted to memory-less strategies. As shown in [AL91], if we are allowed to add to the concrete system auxiliary *history* and *prophecy* variables, then the fair simulation method becomes complete for verifying trace inclusion.

Following [AL91], we allow the concrete system $\mathcal{D}_C$ to be augmented with a set V_H of *history variables* and a set V_P of *prophecy variables*. We assume that the three sets, V_C, V_H, and V_P, are pairwise disjoint. The result is an augmented concrete system $\mathcal{D}_C^* : \langle V_C^*, \Theta_C^*, \rho_C^*, \mathcal{J}_C, \mathcal{C}_C \rangle$, where

$$V_C^* = V_C \cup V_H \cup V_P \qquad \Theta_C^* = \Theta_C \wedge \bigwedge_{x \in V_H} (x = f_x(V_C, V_P))$$
$$\rho_C^* = \rho_C \wedge \bigwedge_{x \in V_H} x' = g_x(V_C^*, V_C', V_P') \wedge \bigwedge_{y \in V_P} y = \varphi_y(V_C)$$

In these definitions, f_x and g_x are state functions, while each $\varphi_y(V_C)$ is a future temporal formula referring only to the variables in V_C. Thus, unlike [AL91], we use *transition relations* to define the values of history variables, and *future* LTL *formulas* to define the values of prophecy variables. The clause $y = \varphi_y(V_C)$ added to the transition relation implies that at any position $j \geq 0$, the value of the boolean variable y is 1 iff the formula $\varphi_y(V_C)$ holds at this position.

The augmentation scheme proposed above is *non-constraining*. Namely, for every computation σ of the original concrete system $\mathcal{D}_C$ there exists a computation σ^* of $\mathcal{D}_C^*$ such that σ and σ^* agree on the values of the variables in V_C.

Handling of the prophecy variables definitions is performed by constructing an appropriate *temporal tester* [KP00b] for each of the future temporal formulas appearing in the prophecy schemes, and composing it with the concrete system.

A similar augmentation of the concrete system has been used in [KPSZ02] in a deductive proof of abstraction, based on [AL91] *abstraction mapping*.

Although fair simulation is verified algorithmically, user intervention is still needed for choosing the appropriate temporal properties to be observed in order to ensure completeness with respect to trace inclusion.

We currently conjecture that we do not really need history variables, and we hope to prove this conjecture in a fuller version of this paper.

6 Examples

Late and Early. Consider, for example, the programs EARLY and LATE in Fig. 2 (graphic representation in Fig. 1). The observable variables are y and z. Wlog, assume that the initial values of all variables are 0. This is a well known example showing the difference between trace inclusion and simulation. Indeed, the two systems have the same set of traces. Either y assumes 1 or y assumes 2. On the other hand, EARLY does not simulate LATE. This is because we do not know whether state $\langle \ell_1,\ x{:}0,\ z{:}1\rangle$ of system LATE should be mapped to state $\langle \ell_1,\ x{:}1,\ z{:}1\rangle$ or state $\langle \ell_1,\ x{:}2,\ z{:}1\rangle$ of system EARLY. Our algorithm shows that indeed EARLY does not simulate LATE.

$$\text{EARLY} :: \begin{bmatrix} \ell_0 : x, z := \{1,2\}, 1 \\ \ell_1 : z := 2 \\ \ell_2 : y, z := x, 3 \end{bmatrix} \qquad \text{LATE} :: \begin{bmatrix} \ell_0 : z := 1 \\ \ell_1 : x, z := \{1,2\}, 2 \\ \ell_2 : y, z := x, 3 \end{bmatrix}$$

Fig. 2. Programs EARLY and LATE.

Since EARLY and LATE have the same set of traces, we should be able to augment LATE with prophecy variables that tell EARLY how to simulate it. In this case, we add a *tester* T_φ for the property $\varphi : \diamondsuit(y = 1)$. The tester introduces a new boolean variable x_φ which is true at a state s iff $s \models \varphi$. Whenever T_φ indicates that LATE will eventually choose $x = 1$, EARLY can safely choose $x = 1$ in the first step. Whenever the tester for $\diamondsuit(y = 1)$ indicates that LATE will never choose $x = 1$, EARLY can safely choose $x = 2$ in the first step. Denote by LATE$^+$ the combination of LATE with the tester $\diamondsuit(y = 1)$. Applying our algorithm to LATE$^+$ and EARLY, indicates that LATE$^+$ $\preceq_f$ EARLY implying $\mathit{Obs}(\text{LATE}) \subseteq \mathit{Obs}(\text{EARLY})$.

Fair Discrete Modules and Open Computations. For the main application of our technique, we need the notions of an *open system* and *open computations*.

We define a *fair discrete module* (FDM) to be a system $M : \langle V, \mathcal{O}, W, \Theta, \rho, \mathcal{J}, \mathcal{C}\rangle$ consisting of the same components as an FDS plus the additional component:

- $W \subseteq V$: A set of *owned* variables. Only the system can modify these variables. All other variables can also be modified by steps of the environment.

An (*open*) *computation* of an FDM M is an infinite sequence $\sigma : s_0, s_1, \ldots$ of V-states which satisfies the requirements of initiality, justice, and compassion as any other FDS, and the requirement of consecution, reformulated as follows:

- *Consecution*: For each $j = 0, 1, ...,$
  - $s_{2j+1}[W] = s_{2j}[W]$. That is, s_{2j+1} and s_{2j} agree on the interpretation of the owned variables W.
  - s_{2j+2} is a ρ-successor of s_{2j+1}.

Thus, an (open) computation of an FDM consists of a strict interleaving of system with environment actions, where the system action has to satisfy ρ, while the environment step is only required to preserve the values of the owned variables.

Two FDM's $\mathcal{D}_1$ and $\mathcal{D}_2$ are compatible if $W_1 \cap W_2 = \emptyset$ and $V_1 \cap V_2 = O_1 \cap O_2$. The asynchronous parallel composition of two compatible FDM's $M = M_1 \parallel M_2$ is defined similarly to the case of composition of two FDS's where, in addition, the owned variables of the newly formed module is obtained as the union of W_{M_1} and W_{M_2}. Module M_2 is said to be a *modular abstraction* of a comparable module M_1, denoted $M_1 \sqsubseteq_M M_2$, if $\mathit{Obs}(M_1) \subseteq \mathit{Obs}(M_2)$. A unique feature of the modular abstraction relation is that it is *compositional*, i.e. $M_1 \sqsubseteq_M M_2$ implies $M_1 \parallel M \sqsubseteq_M M_2 \parallel M$. This compositionality allows us to replace a module M_1 in any context of parallel composition by another module M_2 which forms a modular abstraction of M_1 and obtain an abstraction of the complete system, as needed in the network invariants method.

It is straightforward to reduce the problem of checking modular abstraction between modules to checking abstraction between FDS's using the methods presented in this paper. This reduction is based on a transformation which, for a given FDM $M : \langle V, \mathcal{O}, W, \Theta, \rho, \mathcal{J}, \mathcal{C} \rangle$, constructs an FDS $\mathcal{D}_M : \langle \widetilde{V}, \mathcal{O}, \widetilde{\Theta}, \widetilde{\rho}, \mathcal{J}, \mathcal{C} \rangle$, such that the set of observations of M is equal to the set of observations of $\mathcal{D}_M$. The new components of $\mathcal{D}_M$ are given by:

$$\begin{array}{ll} \widetilde{V} : V \cup \{t : \textbf{boolean}\} & \qquad \widetilde{\Theta} : \Theta \ \wedge \ t \\ \widetilde{\rho} \ : \rho \ \wedge \ \neg t \ \wedge \ t' \quad \vee & \quad \mathit{pres}(W) \ \wedge \ t \ \wedge \ \neg t' \end{array}$$

Thus, system $\mathcal{D}_M$ uses a fresh boolean variable t to encode the turn taking between system and environment transitions.

The Dining Philosophers. As a second example, we consider a solution to the dining philosophers problem.As originally described by Dijkstra, n philosophers are seated around a table, with a fork between each two neighbors. In order to eat a philosopher needs to acquire the forks on both its sides. A solution to the problem consists of protocols to the philosophers (and, possibly, forks) that guarantee that no two adjacent philosophers eat at the same time and that every hungry philosopher eventually eats.

A solution to the dining philosophers is presented in [KPSZ02], in terms of *binary processes*. A binary process $Q(\boldsymbol{x}; \boldsymbol{y})$ is an FDM with two observable variables x and y. Two binary processes Q and R can be composed to yield another binary process, using the *modular composition* operator $\circ$ defined by

$$(Q \circ R)(x; z) \quad = \quad [\textbf{restrict } y \textbf{ in } Q(x; y) \parallel R(y; z)]$$

where **restrict y** is an operator that removes variable y from the set of observable variables and places it in the set of owned variables.

In Fig. 3a we present a chain of n deterministic philosophers, each represented by a binary process $Q(\mathit{left}; \mathit{right})$. This solution is studied in [KPSZ02] as an example of parametric systems, for which we seek a *uniform* verification (i.e. a single verification valid for any n). The uniform verification is presented using the network invariant method, which calls for the identification of a network invariant $\mathcal{I}$ which can safely replace the chain Q^n. The adequacy of the network invariant is verified using an inductive argument which calls for the verification of abstractions. In [KPSZ02] we present a deductive proof to the dining philosophers, based on [AL91] abstraction mapping method, using two different network invariants.

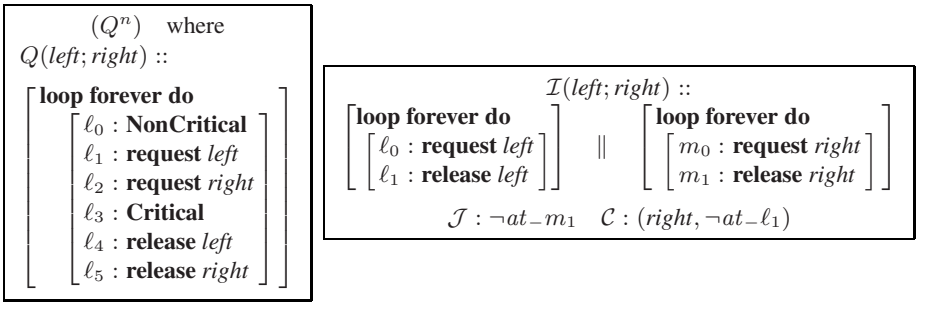

Fig. 3. (a) Program DINE. (b) the two halves abstraction.

Here, we consider the same invariants, and verify all the necessary abstractions using our algorithm for fair simulation. In both cases, no auxiliary variables are needed.

The "*Two-Halves*" Abstraction. The first network invariant $\mathcal{I}(left; right)$ is presented in Fig. 3b and can be viewed as the parallel composition of two "one-sided" philosophers. The compassion requirement reflects the fact that $\mathcal{I}$ can deadlock at location ℓ_1 only if, from some point on, the fork on the right (*right*) is continuously unavailable.

To establish that $\mathcal{I}$ is a network invariant, we verify the abstractions $(Q \circ Q) \sqsubseteq_M \mathcal{I}$ and $(Q \circ \mathcal{I}) \sqsubseteq_M \mathcal{I}$ using the fair simulation algorithm.

The "*Four-by-Three*" Abstraction. An alternative network invariant is obtained by taking $\mathcal{I} = Q^3$, i.e. a chain of 3 philosophers. To prove that this is an invariant, it is sufficient to establish the abstraction $Q^4 \sqsubseteq_M Q^3$, that is, to prove that 3 philosophers can faithfully emulate 4 philosophers.

Experimental Results. We include in our implementation the following steps:

- Removal of all unfeasible states from both systems. Thus, the first step evaluates the set of feasible states for each system [KPR98].
- Recall that fair simulation implies simulation [HKR97]. Let $S \subseteq \Sigma_{cor}$ denote the maximal simulation relation. To optimize the algorithm we further restrict player A's moves to S instead of Σ_{cor}.

The following summarizes the running time for some of our experiments.

$(Q \circ Q) \sqsubseteq_M \mathcal{I}$	44 secs.	$(Q \circ \mathcal{I}) \sqsubseteq_M \mathcal{I}$	6 secs.	$Q^4 \sqsubseteq_M Q^3$	178 secs.

7 Acknowledgments

We thank Orna Kupferman for suggesting using fair simulation for algorithmically verifying abstraction of reactive systems.

References

[AL91] M. Abadi and L. Lamport. The existence of refinement mappings. *TCS*, 82, 1991.

[BR96] B. Bloom and R.Paige. Transformational design and implementation of a new efficient solution to the ready simulation problem. *SCP*, 24:189–220, 1996.

[Cho74] Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *JCSS*, 1974.

[dAHM01] L. de Alfaro, T.A. Henzinger, and R. Majumdar. From verification to control: dynamic programs for omega-regular objectives. In *16th LICS*, 2001.

[EL86] E. A. Emerson and C. L. Lei. Efficient model-checking in fragments of the propositional modal μ-calculus. In *1st LICS*, 267–278, 1986.

[Eme97] E.A. Emerson. Model checking and the μ-calculus. In *Descriptive Complexity and Finite Models*, pages 185–214. AMS, 1997.

[GH82] Y. Gurevich and L.A. Harrington. Automata, trees and games. In *14th STOC*, 1982.

[GL94] O. Grumberg and D.E. Long. Model checking and modular verification. *TOPLAS*, 16(3):843–871, 1994.

[HHK95] M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *36th FOCS*, 453–462, 1995.

[HKR97] T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. In *8th CONCUR*, LNCS 1243, 273–287, 1997.

[HR00] T. Henzinger and S. Rajamani. Fair bisimulation. In *6th TACAS* LNCS 1785, 2000.

[Jur00] M. Jurdzinski. Small progress measures for solving parity games. In *17th STACS*, LNCS 1770, 290–301, 2000.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *TCS*, 27:333–354, 1983.

[KP00a] Y. Kesten and A. Pnueli. Control and data abstractions: The cornerstones of practical formal verification. *STTT*, 2(1):328–342, 2000.

[KP00b] Y. Kesten and A. Pnueli. Verification by finitary abstraction. *IC*, 163:203–243, 2000.

[KPR98] Y. Kesten, A. Pnueli, and L. Raviv. Algorithmic verification of linear temporal logic specifications. In *25th ICALP*, LNCS 1443, 1–16. 1998.

[KPSZ02] Y. Kesten, A. Pnueli, E. Shahar, and L. Zuck. Network invariants in action. In *13th CONCUR*, LNCS 2421, 101–105, 2002.

[KPV00] O. Kupferman, N. Piterman, and M.Y. Vardi. Fair equivalence relations. In *20th FSTTCS*, LNCS 1974, 151–163. 2000.

[KV98] O. Kupferman and M.Y. Vardi. Weak alternating automata and tree automata emptiness. In *30th STOC*, 224–233, 1998.

[LT87] K. Lodaya and P.S. Thiagarajan. A modal logic for a subclass of events structures. In *14th ICALP*, LNCS 267, 290–303, 1987.

[MAB+94] Z. Manna, A. Anuchitanukul, N. Bjørner, A. Browne, E. Chang, M. Colón, L. De Alfaro, H. Devarajan, H. Sipma, and T.E. Uribe. STeP. Tech. Report, Stanford 1994.

[Mil71] R. Milner. An algebraic definition of simulation between programs. *IJCAI*, 1971.

[MP95] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, New York, 1995.

[Saf92] S. Safra. Exponential determinization for ω-automata with strong-fairness acceptance condition. In *24th STOC*, 1992.

[Var91] M. Y. Vardi. Verification of concurrent programs – the automata-theoretic framework. *APAL*, 51:79–98, 1991.

[WL89] P. Wolper and V. Lovinfosse. Verifying properties of large sets of processes with network invariants. In *AVMFS*, LNCS 407, 68–80. 1989.