A hierarchy of polynomial-time computable simulations for automata

Kousha Etessami

Bell Labs, Murray Hill, NJ
email: kousha@research.bell-labs.com

Abstract. We define and provide algorithms for computing a natural hierarchy of simulation relations on the state-spaces of ordinary transition systems, finite automata, and Büchi automata. These simulations enrich ordinary simulation and can be used to obtain greater reduction in the size of automata by computing the automaton quotient with respect to their underlying equivalence. State reduction for Büchi automata is useful for making explicit-state model checking run faster ([EH00,SB00,EWS01]).
We define k-simulations, where 1-simulation corresponds to ordinary simulation and its variants for Büchi automata ([HKR97,EWS01]), and k-simulations, for $k > 1$, generalize the game definition of 1-simulation by allowing the Duplicator to use k pebbles instead of 1 (to "hedge its bets") in response to the Spoiler's move of a single pebble. As k increases, k-simulations are monotonically non-decreasing relations. Indeed, when k reaches n, the number of states of the automaton, the n-simulations defined for finite-automata and for labeled transition systems correspond precisely to language containment and trace containment, respectively. But for each fixed k, the maximal k-simulation relation is computable in polynomial time: $n^{O(k)}$.
This provides a mechanism with which to trade off increased computing time for larger simulation relation size, and more potential reduction in automaton size. We provide algorithms for computing k-simulations using a natural generalization of a prior efficient algorithm based on parity games ([EWS01]) for computing various simulations. Lastly, we observe the relationship between k-simulations and a k-variable interpretation of modal logic.

1 Introduction

Computing simulation relationships between the states of transition systems (Kripke structures), NFAs, and Büchi automata is useful for several purposes. One purpose is to efficiently establish language containment between automata and/or to establish the preservation of properties from fragments of various modal logics. Another important use of simulation is to reduce the state space of an automaton while preserving the underlying language it accepts. This is done by computing the quotient of the automaton with respect to the equivalence underlying a simulation preorder. Smaller Büchi automata obtained in such a manner can be used in the standard algorithms for explicit-state model checking based on Büchi automata ([VW86]), like those used in the tool SPIN ([Hol97]), to make the model checking computation more efficient (see, e.g., [EH00,SB00]).

Thus, in order to obtain the greatest possible state reduction, it is desirable to have as large a "simulation relation" as possible, which still has a quotient that preserves the language of the automaton, and which still remains efficiently computable. Finding the optimal nondeterministic automaton, or computing language containment in general, are well-known PSPACE-hard problems, so one can only hope for efficient solutions that achieve substantial size reduction in practice.

In prior work by this author together with Wilke and Schuller [EWS01], we provided efficient algorithms for computing a variety of simulations (and bisimulations) on the state-space of Büchi automata. These simulations included among them *fair* simulation, introduced by Henzinger, et. al. ([HKR97]), but we also introduced a new notion of simulation called *delayed* simulation, and showed that while *fair* simulation quotients do not preserve the language of an automaton, delayed simulation quotients do and yet they can be arbitrarily coarser (i.e., larger) than standard direct simulation, used in the past for quotienting automata. Our algorithms for computing these simulations used a parity game framework, employing an algorithm by Jurdzinski ([Jur00]) for computing the winner in a parity game. These algorithms have been implemented, and an executable for an optimized translation from temporal logic to Büchi automata using them is available[1].

In this paper we vastly generalize the game-theoretic simulation framework, providing a natural hierarchy of ever-coarser polynomial time computable simulations for labeled transition systems (LTSs), NFAs, and Büchi automata. In the setting of finite automata, these k-simulations allow the Duplicator in the simulation game to perform a kind of partial subset construction, restricted to subsets of size at most k, as it tries to "match" the moves of the Spoiler on nondeterministic automata. Indeed, in the finite automaton case, if n is the number of states of an automaton, then n-simulation corresponds precisely to finite language containment. This is so because subsets of size n suffice to perform the full-fledged subset construction. The situation in the Büchi automaton case is more subtle, but there as well k-simulations can be appropriately defined to generalize the notions of fair and delayed simulation. k-simulations thus provide a parameterized way to trade off the use of more computing resources to obtain larger simulations and hence improved reduction in automaton size via quotients.

The k-simulation games we define are related to, but markedly different from, well-known Ehrenfeucht-Fraisse games for, e.g., k-variable first-order logic (see, e.g., [IK89]). The most important difference is that k-simulation games are peculiarly asymmetric: the Duplicator controls a different number of pebbles than the Spoiler.

[1] `http://cm.bell-labs.com/who/kousha`.

We also observe how k-simulations are naturally related to k-variable interpretations of modal logic and the μ-calculus. This characterization is analogous to the well-known fact that states in a transition system are bisimilar iff they agree on all modal logic properties ([HM85]).

In recent research, simulation quotients have been used for a variety of purposes, including, e.g., indexing and describing the structure of XML data ([ABS99,KSBG02]). We envision that k-simulations, for small k, can be used as richer notions of simulation in any setting where some extra computing time is affordable in order to obtain a larger simulation relation on the state space. This will however be more practical and plausible if the space bounds of our algorithms can be improved upon.

In the next section we present background and definitions, and in the two subsequent sections we prove the desirable properties of k-simulations and we describe the algorithmic framework for computing them using parity games. In the last section we describe the connection to modal logic interpreted over k variables.

2 Background and Definitions

We will not review the standard definition of simulation and its variants incorporating acceptance notions (see [EWS01,HKR97]). Instead we go directly to the general definition of k-simulations and point out how specializing to $k = 1$ naturally yields the standard simulations. We then motivate the general definition with examples. All definitions are presented from a game-theoretic viewpoint.

Our definitions focus on Büchi automata, but many results are relevant for finite automata as well as state machines without acceptance criteria, i.e., labeled transition systems (LTSs). Recall that a Büchi automaton $A = \langle \Sigma, Q, q_I, \Delta, F\rangle$ has an alphabet Σ, a state set Q, an initial state $q_I \in Q$, a transition relation $\Delta \subseteq Q \times \Sigma \times Q$, and a set of final states $F \subseteq Q$. We will henceforth assume that a Büchi automaton has no *dead ends*: from each state there is a path of length ≥ 1 to *some* state in F. It is easy to assure this property without changing the accepting runs from any state, using a simple search to eliminate unnecessary states and transitions.

Recall that a *run* of A is a sequence $\pi = q_0 a_0 q_1 a_1 q_2 \dots$ of states alternating with letters such that for all i, $(q_i, a_i, q_{i+1}) \in \Delta$. The ω-word associated with π is $w_\pi = a_0 a_1 a_2 \dots$ The run π is *initial* if it starts with q_I; it is *accepting* if there exist infinitely many i with $q_i \in F$. The ω-language defined by A is $L_\omega(A) = \{w_\pi \mid \pi \text{ is an initial, accepting run of } A\}$. When we ignore the accepting states F, A can be viewed a basic LTS where every infinite run is considered an accepting run. The set of traces of A, denoted $Tr(A) = \{w_\pi \mid \pi \text{ is an initial run of } A\}$, constitutes all words over which a run exists. Finally, when A is viewed as an ordinary NFA, the language of finite strings it accepts is denoted $L(A)$. We may want to change the start state of A to a different state q; the revised automaton will be denoted by $A[q]$. We are ready to define the k-simulation game.

Given a Büchi automaton A, $q_0 \in Q$, and a k-tuple[2] (a.k.a., k-vector) $\mathbf{q}'_0 = (q'_{0,1}, \dots, q'_{0,k}) \in Q^k$ we define the *k-simulation* game $G_A(q_0, \mathbf{q}'_0)$. The game is played by two players, *Spoiler* and *Duplicator,* in rounds. Initially, round 0, a pebble, *Red*, is placed on q_0, and k pebbles $(Blue_1, Blue_2, \dots, Blue_k)$ are placed, respectively, on $q'_{0,1}, \dots, q'_{0,k}$. We will refer to this vector of blue pebbles as *Blue*. We define a transition relation $\Gamma_{A,k}$ on k-vectors as follows (we write Γ_A when k is apparent from the context). Given $\mathbf{q}_0 = (q_{0,1}, \dots, q_{0,k})$ and $\mathbf{q}'_0 = (q'_{0,1}, \dots, q'_{0,k})$, $(\mathbf{q}_0, a, \mathbf{q}'_0) \in \Gamma_A$ iff for each $i \in [k]$ there is some $j \in [k]$ such that $(q_{0,j}, a, q'_{0,i}) \in \Delta$. Note that i and j need not be the same! Assume that at the beginning of round i *Red* is on state q_i and *Blue* is on $\mathbf{q}'_i$. Then:

1. Spoiler chooses a transition $(q_i, a, q_{i+1}) \in \Delta$ and moves *Red* to q_{i+1}.
2. Duplicator, responding, chooses a transition $(\mathbf{q}'_i, a, \mathbf{q}'_{i+1}) \in \Gamma_A$ and moves the vector of *Blue* pebbles to $\mathbf{q}'_{i+1}$. If no such transition exists (i.e., Duplicator can't move) then the game halts and Spoiler wins.

Either the game halts, in which case Spoiler wins, or the game produces two infinite sequences: a run $\pi = q_0 a_0 q_1 a_1 q_2 \dots$ and a sequence $\pi' = \mathbf{q}'_0 a_0 \mathbf{q}'_1 a_1 \mathbf{q}'_2 \dots$, built from the transitions taken by the *Red* pebble and the vector of *Blue* pebbles. Given these infinite sequences, there are several ways one can define the winner of the game. Each determines a different notion of k-simulation. Two of the basic notions related to ordinary transition systems and to finite automata, respectively, are as follows:

1. *Ordinary k-simulation game:* denoted $\mathbf{G}^o_A(q_0, \mathbf{q}'_0)$. Duplicator wins, regardless. (In other words, acceptance conditions are ignored; Duplicator wins as long as the game does not halt).
2. *Direct k-simulation game:* denoted $\mathbf{G}^{di}_A(q_0, \mathbf{q}'_0)$. Duplicator wins iff, for all i, if $q_i \in F$, then also $q'_{i,j} \in F$ for <u>some</u> $j \in \{1, \dots, k\}$.

We will need some definitions in order to present the other notions of k-simulation, designed for Büchi automata. We want to define what it means for a vector of states $\mathbf{q}'_i$ to be *good* since some prior round, which roughly speaking is analogous to a single state q being an accept state.

The vector $\mathbf{q}'_i$ defines the location of all blue pebbles at round i. We will first define what it means for a pebble $Blue_j$ at some round m to have *"seen"* a state q since some prior round i. As a base case, we say $Blue_j$ at round $m = i$ has seen state q since round i if indeed $q'_{i,j} = q$. Inductively, $Blue_j$, at round $m > i$ has seen state q since round i if either $q'_{m,j} = q$ or if there is some r such that $(q'_{m-1,r}, a_{m-1}, q'_{m,j}) \in \Delta$, and such that $Blue_r$ at round $m-1$ has seen state q since round i.

[2] We have chosen to describe the states covered by Blue pebbles as a tuple rather than a set. This is done only to make some parts of the presentation more convenient, and in fact the only information from a tuple $\mathbf{q}'$ that matters for all our simulations is the set of states in it; even the mapping of pebbles to states can be taken to be a canonical mapping. This will be described further in the proof of theorem 3 which appears in the appendix.

Now to define goodness of a vector at round m: $\mathbf{q}'_m$ is *good* since round $m' \leq m$ if at round m every pebble $Blue_j$ has seen some accept state since round m' and m is the least round for which this holds. Now we can define the notions of simulation designed for Büchi automata:

3. *Delayed k-simulation game:* denoted $\mathbf{G}^{de}_A(q_0, \mathbf{q}'_0)$. Duplicator wins iff, for all i, if $q_i \in F$, then there exists $j \geq i$ such that $\mathbf{q}'_j$ is good since round i.
4. *Fair k-simulation game:* denoted $\mathbf{G}^{f}_A(q_0, \mathbf{q}'_0)$. Duplicator wins iff $\forall\, i\ \exists\, j \geq i$ such that $\mathbf{q}'_j$ is good since round i, or else there are only finitely many i such that $q_i \in F$ (in other words, if there are infinitely many i such that $q_i \in F$, then for each such i, there exists $j \geq i$ such that $\mathbf{q}'_j$ is good since round i).

Given these notions for simulation games, we now define what it means for a state vector $\mathbf{q}'$ to k-simulate a state q, and using this we also define what it means for a single state q' to k-simulate another state q. We will speak of strategy and winning strategy without formally defining them here. These are standard game theoretic notions (see [EWS01]).

Definition 1. *For an automaton A,*

1. *For each simulation type $\tau \in \{$*ordinary, direct, delayed, fair*$\}$, we say k-vector $\mathbf{q}'$ τ-k-simulates state q, denoted $q \sqsubseteq^\tau_k \mathbf{q}'$, if Duplicator has a winning strategy in $G^\tau_A(q, \mathbf{q}')$.*
2. *For a single state q', we say q' τ-k-simulates q if $\mathbf{q}' = (q', q', \ldots, q')$ τ-k-simulates q. We overload $\sqsubseteq^\tau_k$ to denote the binary τ-k-simulation relation on states of A. We denote by $\preceq^\tau_k$ the transitive closure of the binary relation $\sqsubseteq^\tau_k$.*
3. *We define $q \approx^\tau_k q'$ to hold if and only if $q \preceq^\tau_k q'$ and $q' \preceq^\tau_k q$.*

It is not hard to show that for $k > 1$, $\sqsubseteq^\tau_k$ itself is not in general a transitive relation. That is why we define $\preceq^\tau_k$ as the transitive closure. Consequently, $\approx^\tau_k$ is an equivalence. If we restrict attention to 1-simulations, the definition of "goodness" for a 1-vector amounts to nothing more than the state being an accept state, and hence all variants of 1-simulation amount to the same variants of standard simulation given in [EWS01].

To begin to understand the motivation for k-simulations with $k > 1$, consider the automata depicted in Figure 1.

The state q'_0 obviously does not simulate the state q_0. This is because there must be a way for Duplicator to counter a move by Spoiler's *Red* pebble from q_0 to q_1. However, Duplicator, with its single *Blue* pebble, has to choose either to move from q'_0 to q'_1 or to q'_2, and with either choice the Spoiler defeats her in the next move. On the other hand, if Duplicator had two Blue pebbles with which to "hedge its bets" it could move one pebble to q'_1 and the other to q'_2, and in so doing be prepared for any subsequent move by Spoiler. It is trivial to generalize these automata so that exactly k pebbles are necessary and sufficient, rather than 2, for duplicator to be able to win the game. From the definitions, using the example in figure 1, and the examples given in [EWS01], we can establish the following:

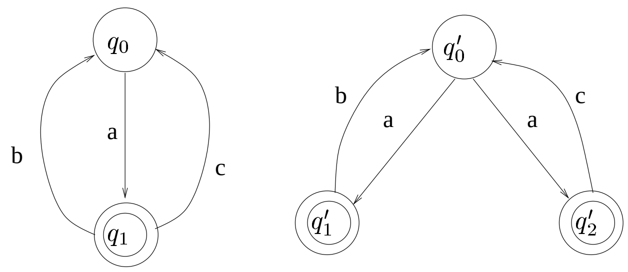

Fig. 1. q'_0 2-simulates q_0 but does not 1-simulate it.

Theorem 1. *For $k \geq 1$, $\tau \in \{o, di, de, f\}$, for any A, the following containments hold:*

1. *$\preceq_k^\tau \subseteq \preceq_{k+1}^\tau$. Furthermore, $\forall k$, there are A over which the containment is strict.*
2. *$\preceq_k^{di} \subseteq \preceq_k^o$, and $\preceq_k^{de} \subseteq \preceq_k^f \subseteq \preceq_k^o$. Furthermore, for each k there are automata over which all these containments are strict ([EWS01]).*

The example of Figure 1 is a textbook case of why standard simulation is too weak to capture language containment. However, whereas it is PSPACE-complete to compute language containment, as we will see it will only require polynomial time to compute 2-simulations, or k-simulations for any fixed k. Quotients of automata are defined as follows:

Definition 2. For a Büchi automaton A, and an equivalence relation $\approx$ on the states of A, let $[q]$ denote the equivalence class of $q \in Q$ with respect to $\approx$. The *quotient* of A with respect to $\approx$ is the automaton $A/\!\approx\, = \langle \Sigma, Q/\!\approx, \Delta_\approx, [q_I], F_\approx \rangle$ where $\Delta_\approx = \{([q], a, [q']) \mid \exists\, q_0 \in [q],\ q'_0 \in [q'],$ such that $(q_0, a, q'_0) \in \Delta\}$, and where $F_\approx = \{[q] \mid \exists\, q_0 \in [q]$ such that $q_0 \in F\}$.

3 Useful properties of k-simulations

In this section we show some basic properties of k-simulations that make them useful for checking language containment and for state space reduction.

Theorem 2. *For $k \geq 1$, for $q, q' \in Q$, where $|Q| = n$:*

1. *If $q \preceq_k^o q'$ then $Tr(A[q]) \subseteq Tr(A[q'])$. Moreover, $q \preceq_n^o q'$ iff $Tr(A[q]) \subseteq Tr(A[q'])$.*
2. *If $q \preceq_k^{di} q'$ then $L(A[q]) \subseteq L(A[q'])$. Furthermore $q \preceq_n^{di} q'$ if and only if $L(A[q]) \subseteq L(A[q'])$.*
3. *For $\tau \in \{de, f\}$: if $q \preceq_k^\tau q'$ then $L_\omega(A[q]) \subseteq L_\omega(A[q'])$.*

Proof. For part 1 and 2, the claim that if $q \preceq_k^{di(o)} q'$ then $L(A[q]) \subseteq L(A[q'])$ ($Tr(A[q]) \subseteq Tr(A[q'])$) follows readily from definitions. The converse for $k = n$, namely that if $L(A[q]) \subseteq L(A[q'])$ then $q \preceq_n^{di} q'$, holds because the Duplicator's strategy will be the standard subset construction: at each round the Duplicator will maintain a pebble on each of the at most n possible states that one could reach from q' using the prefix of the string traversed so far. It is easily seen that the subset construction can also be used to determinize LTSs, and hence $Tr(A[q]) \subseteq Tr(A[q'])$ implies that $q \preceq_n^o q'$.

For part 3, it suffices to prove the statement for $\tau = f$ because by the second part of Proposition 1, $\preceq_k^f$ is larger than $\preceq_k^{de}$. Since set containment is transitive, all we need to show is that if $q \sqsubseteq_k^f q'$ then $L_\omega(A[q]) \subseteq L_\omega(A[q'])$. Suppose $q \sqsubseteq_k^f q'$, and let $\pi = qa_0q_1a_1\ldots$ be an accepting run of Büchi automaton $A[q]$. We want to find an accepting run of $A[q']$ over the same word $w_\pi = a_0a_1\ldots$. We know from Duplicator's winning strategy in $G_A^f(q, \mathbf{q}')$ that there exists a sequence $\pi' = \mathbf{q}'a_0\mathbf{q}'_1a_1\ldots$ that witnesses Duplicator's win if the Spoiler's strategy is to play the run π. We will construct an accepting run of $A[q']$ over w using π'. We know that for infinitely many rounds, $i_1 < i_2 < i_3 < \ldots$, $\mathbf{q}'_{i_j}$ is good in π' since prior round $i_{j-1} < i_j$ (setting $i_0 = 0$). By induction on j, we can construct a run such that the run's prefix of length i_j traverses an accept state at least j times. □

Next, we show that quotienting with respect to delayed k-simulation preserves the ω-language of automaton A, while quotienting with respect to direct k-simulation preserves the finite language. As noted in [EWS01], fair (1-)simulation quotients do not preserve the ω-language of A. Also, as we will see in Proposition 2, unlike direct 1-simulation, direct k-simulation does not preserve the ω-language either.

Proposition 1. *Given a finite automaton A, $L(A/\approx_k^{di}) = L(A)$.*

Proof. We simply observe that the largest direct simulation equivalence, $\approx_n^{di}$, where $n = |A|$, is the language equivalence relation, i.e., $q \approx_n^{di} q'$ iff $L(A[q]) = L(A[q'])$. It is known and easy to show that quotienting a finite automaton with respect to finite language equivalence does not alter the (finite) language of the automaton. Since $\approx_k^{di}$ is a refinement of language equivalence, the claim follows. □

We now come to a theorem which justifies our definitions for the purpose of state space reduction on Büchi automata.

Theorem 3. *For a Büchi automaton A, $L_\omega(A/\approx_k^{de}) = L_\omega(A)$.*

The proof is involved and is given in the appendix. We now observe that finite language equivalence quotients, and direct k-simulation equivalence quotients, do not preserve ω-languages. This is in contrast to the case of direct 1-simulation ([EH00,SB00]). (We could have defined direct k-simulation is such a way that duplicator must respond to a move to an accept state with a move to a vector *all* of whose states are accept states, making direct k-simulation strictly weaker than delayed k-simulation. But that definition would have the disadvantage that direct k-simulations would not reach finite language containment as k increases to n.)

Proposition 2. *There are Büchi automata A such that $L_\omega(A/\approx_2^{di}) \neq L_\omega(A)$.* (Hence quotienting with respect to finite language equivalence, i.e., $\approx_{|A|}^{di}$, also does not preserve the ω-language of A.)

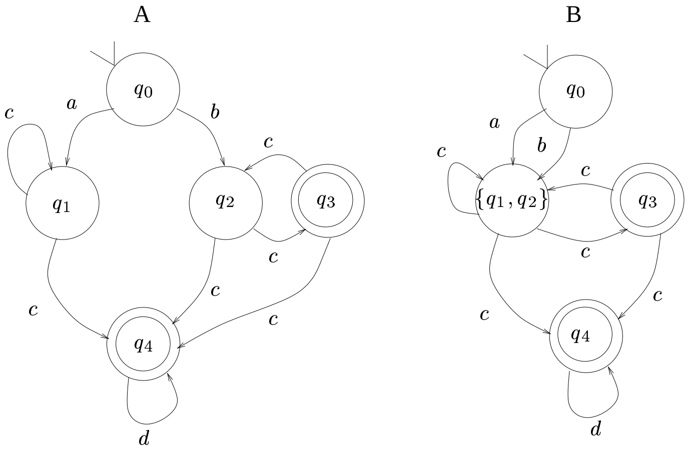

Fig. 2. $A/\approx_2^{di} = B$

Proof. Consider the automaton A depicted on the left of Figure 2. The finite language accepted starting at states q_1 and q_2 is identical: c^+d^*. In other words, $q_1 \approx_5^{di} q_2$, and in fact $q_1 \approx_2^{di} q_2$. Hence, the quotient $A/\approx_2^{di}$ is given by the automaton B on the right of Figure 2. However, q_1 and q_2 do not share the same ω-language: $ac^\omega \in L_\omega(B) \setminus L_\omega(A)$. □

4 Computing k-simulations

Our algorithms for computing τ-k-simulations are based on the same framework as the algorithms for computing τ-1-simulations described in [EWS01]. We reduce each simulation computation to a parity game computation, by building an appropriate parity game graph $\mathbf{G}_A^{\tau,k}$, such that q' τ-k-simulates q if an only if Player Zero has a winning strategy from a corresponding node $v_{q,q'}$ in the game graph $\mathbf{G}_A^{\tau,k}$. These parity game graphs have nodes labeled by at most 3 distinct priorities. We then apply an algorithm by Jurdzinski ([Jur00]), along with some enhancements described in [EWS01] which allow us to achieve the desired time and space bounds, to compute the winning nodes in the parity games and thus to compute the simulation relation.

We will not review in full the definition of parity games. See [Jur00,EWS01] for notation that matches ours. A parity game graph $\mathbf{G} = \langle V_0, V_1, E, p \rangle$ has two disjoint sets of vertices, V_0 and V_1 (nodes where it is player *Zero* and player *One*'s turn to play, respectively), whose union is denoted V. There is an edge set $E \subseteq V \times V$, and $p\colon V \to \{0, \ldots, d-1\}$ is a mapping that assigns a *priority* to each vertex.

Because these constructions are quite similar to those given in [EWS01], we only describe the most complicated parity game graph here, corresponding to delayed k-simulation, $\mathbf{G}_A^{de,k} = \langle V_0^{de,k}, V_1^{de,k}, E^{de,k}, p^{de,k} \rangle$. Please see [EWS01] for several examples of how parity game graphs are constructed from simulation games. For each state $q \in Q$ and each k-vector $\mathbf{q}' \in Q^k$, each binary bit b, and each binary k-vector $\overline{b'}$ there will be a vertex $v_{(q,\mathbf{q}',b,\overline{b'})} \in V_1^{de,k}$. The bit b will be used to record whether, thus far in the simulation game, the *Red* pebble has seen an accept state without the vector of *Blue* pebbles having been on a good vector since then. The vector $\overline{b'}$ will be used to record, for each pebble $Blue_i$ whether it has seen an accept state since the last time *Red* was on an accept state. The vertices $V_0^{de,k}$ will be similar, but each node $v_{(q,\mathbf{q}',b,\overline{b'},a)}$ will also record the label a of the last transition taken by the *Red* pebble in spoiler's move. The vertices of $\mathbf{G}_A^{de,k}$ are thus as follows:

$$\begin{aligned} V_1^{de,k} &= \{v_{(q,\mathbf{q}',b,\overline{b'})} \mid q \in Q,\ \mathbf{q}' \in Q^k,\ b \in \{0,1\},\ \overline{b'} \in \{0,1\}^k\} \ , \\ V_0^{de,k} &= \{v_{(q,\mathbf{q}',b,\overline{b'},a)} \mid \exists q''((q'',a,q) \in \Delta)\} \ . \end{aligned}$$

The edges of the game graph reflect the possible moves of spoiler and duplicator in the k-simulation game. Thus edge $(v_{(q,\mathbf{q}',b,\overline{b'})}, v_{(q'',\mathbf{q}',b'',\overline{b'''},a)}) \in E^{de,k}$ corresponds to a move by spoiler of the *Red* pebble from q to q'' using a transition $(q,a,q'') \in \Delta$, while *Blue* is on $\mathbf{q}'$. The bit b is adjusted in b'', so that if q'' is an accept state then b'' is set to 1. Likewise, the bits $\overline{b'}$ are adjusted in $\overline{b'''}$, so that if q'' is an accept state then $\overline{b'''}$ is reset to the all 0 vector. Here, a bit of $\overline{b'''}$ will be set to 1 if the corresponding blue pebble has seen an accept state since the last time *Red* saw an accept state. When all these bits have been set to 1, the bit b will be reset to 0. The edges from vertices in $V_0^{de,k}$ to those in $V_1^{de,k}$ are described similarly, using the appropriate rules that dictate how the bits b and $\overline{b}$ should be modified. The priorities $p^{de,k}(v)$ will be assigned as follows: nodes $v_{(q,\mathbf{q}',b,\bar{b}')}$ in V_1 will get priority b, while nodes in V_0 will get priority 2.

The game graph is such that Zero has a winning strategy from $v_{(q,\mathbf{q}',b,\bar{0})}$ if and only if $\mathbf{q}'$ delayed-k-simulates q, where $b = 1$ if $q \in F$ and $\mathbf{q}'$ is not a vector of final states, and $b = 0$ otherwise. $\mathbf{G}_A^{de,k}$ has $O(2^k n^k m)$ vertices, where m is the number of transitions of A, and it has at most $O(n^{2k-1} 2^{2k} m)$ edges. Using the parity game algorithms described in [Jur00,EWS01], and parity game constructions for the other k-simulations, we get:

Theorem 4. *Given a Büchi automaton A with n states and m transitions:*

- *The relations $\sqsubseteq_k^{de}$ and $\sqsubseteq_k^{f}$ can each be computed in time $O(n^{3k}2^{2k}m)$ and space $O(n^{2k-1}2^{2k}m)$.*
- *The relations $\sqsubseteq_k^{di}$ and $\sqsubseteq_k^{o}$ can both be computed in time and space $O(n^{2k-1}m)$.*

To compute $\preceq_k^{\tau}$, we compute the transitive closure of $\sqsubseteq_k^{\tau}$. Note $\sqsubseteq_1^{\tau}$ is already transitive ([EWS01]).

5 *k*-simulations and modal logic

It is well known that finite transition systems are bisimilar iff they satisfy the same propositional modal logic formulas ([HM85]). More generally, the same is true for modal μ-calculus formulas. Related observations regarding fair simulation and bisimulation and fragments of alternation-free μ-calculus have been made in [HKR97,HR00], who introduced an interpretation of alternation-free μ-calculus over fair transition systems. We observe here the analogous relationship between k-simulations and a k-variable interpretation of modal logic and μ-calculus. Consider the basic propositional modal μ-calculus, a logic whose formulas $\mathcal{L}_\mu$ are generated by the following (redundant) grammar:

$$\Phi ::= true \mid \neg\Phi \mid \Phi \vee \Phi \mid \Phi \wedge \Phi \mid \langle a \rangle \Phi \mid X \mid \mu X.\Phi \mid \nu X.\Phi$$

Here a can be any symbol of the alphabet Σ, and X can be any variable from a set $\mathcal{X}$ of propositional variables. The grammar is given redundantly so we can consider several fragments of $\mathcal{L}_\mu$. First, if we restrict the grammar to disallow the three last forms: X, $\mu X.\Phi$, and $\nu X.\Phi$, we obtain ordinary propositional modal logic, $\mathcal{L}_M$. We consider the restriction of $\mathcal{L}_M$ to its existential fragment, $\exists\mathcal{L}_M$, where negation is eliminated, except only when applied to *true*. Also, we consider a similar restriction of $\mathcal{L}_\mu$ to its existential fragment, $\exists\mathcal{L}_\mu$, where again, negation is eliminated except when applied to *true*. Recall the interpretation of a $\mathcal{L}_M$ formula on a transition system $A = (\Sigma, Q, \Delta)$ and distinguished state $q \in Q$ is as follows. $(A, q) \models true$ always holds, boolean combinations are interpreted in the obvious way, and: $(A, q) \models \langle a \rangle \varphi$ iff $\exists q'$ s.t. $(q, a, q') \in \Delta$ and $(A, q') \models \varphi$. The interpretation of $\mathcal{L}_\mu$ follows the usual fixed point semantic which we will not repeat here. We introduce the following k-variable interpretation of formulas in $\mathcal{L}_M$, derived from the definition of k-simulation. Given $(A, q_1, \ldots, q_k)$, where $\mathbf{q} = (q_1, \ldots, q_k) \in Q^k$: again $(A, q_1, \ldots, q_k) \models^k true$, boolean combinations are obvious, and: $(A, q_1, \ldots, q_k) \models^k \langle a \rangle \varphi$ iff $\exists \mathbf{q}' = (q'_1 \ldots q'_k)$ such that, $(\mathbf{q}, a, \mathbf{q}') \in \Gamma_A$ and $(A, q'_1, \ldots, q'_k) \models^k \varphi$. The simple relationship to k-simulations, analogous to the Henessey-Milner characterization of bisimulations ([HM85]), is as follows:

Proposition 3. *For any finite transition system A, $q \in Q$, $\mathbf{q}' = (q'_1, \ldots, q'_k) \in Q^k$, TFAE:*

1. $q \sqsubseteq_k^{o} \mathbf{q}'$
2. *For every $\exists\mathcal{L}_M$ formula φ, if $(A, q) \models \varphi$ then $(A, \mathbf{q}') \models^k \varphi$.*
3. *For every $\exists\mathcal{L}_\mu$ formula ψ, if $(A, q) \models \psi$ then $(A, \mathbf{q}') \models^k \psi$.*

To generalize this to fair structures would first require a suitable k-variable interpretation of alternation-free modal μ-calculus over automata with fairness constraints, analogous to the single variable interpretation defined in [HKR97]. We do not attempt to provide such a definition here.

6 Conclusions

We have shown how simulation can be understood as the first step in a natural progression towards trace containment and language containment for labeled transition systems and finite automata. We have similarly generalized definitions of fair and delayed simulation for Büchi automata to a progression of ever richer k-simulations; these k-simulations, by contrast, do not "reach" ω-language containment as k grows. This is unavoidable if we wish to be able to use quotients for state space reduction, because quotienting with respect to ω-language equivalence itself does not preserve the ω-language of a Büchi automaton. For the purpose of state reduction we have defined the k-simulation variant of delayed simulation, and shown that it does preserve the ω-language upon quotienting. It will be interesting to find an alternative to our definition of fair k-simulation which does "reach" ω-language containment. Such a simulation likely must more closely mimic Safra's construction [Saf88], and since that construction requires Rabin acceptance conditions, it is not likely that the game graph for it will be encodable with 3-priority parity games like our definition.

Our algorithms for computing k-simulations, based on parity games, can be implemented atop the same parity game framework for computing simulations ([EWS01]) that has already been implemented in our Temporal Massage Parlor platform for generating optimized Büchi automata from temporal logic formulas[3]. However, without substantial heuristic optimization the algorithms are inefficient as k grows, because of k's role in the exponents of both the running time and space. Particularly problematic is the space inefficiency of the algorithms, but there is some reason to believe that the worst case space efficiency of the algorithms can be improved. One obvious optimization which mildly mitigates the space inefficiency is to use, rather than k-vectors, k-sets as described in the appendix, to define the position of the *Blue* pebbles at any given round.

Acknowledgement. Thanks to Thomas Wilke for helpful discussions.

References

[ABS99] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web.* Morgan Kaufmann, 1999.

[EH00] K. Etessami and G. Holzmann. Optimizing Büchi automata. In *Proc. of 11th Int. Conf on Concurrency Theory (CONCUR)*, pages 153–167, 2000.

[3] `http://cm.bell-labs.com/kousha`

[EWS01] K. Etessami, T. Wilke, and R. Schuller. Fair simulation relations, parity games, and state space reduction for Büchi automata. In *Proc. of ICALP'2001*, pages 694–707, 2001.
[HKR97] T. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. In *Proc. of 9th Int. Conf. on Concurrency Theory (CONCUR'97)*, number 1243 in LNCS, pages 273–287, 1997.
[HM85] M. Hennessey and R. Milner. Algebraic laws for nondeterminism and concurrency. *J. of the ACM*, 32(1):137–161, 1985.
[Hol97] G. J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.
[HR00] T. Henzinger and S. Rajamani. Fair bisimulation. In *6th Int. Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS'2000)*, LNCS 1785, pages 299–314, 2000.
[IK89] N. Immerman and K. Kozen. Definability with bounded number of bound variables. *Inform. and Comput.*, 83:121–139, 1989.
[Jur00] M. Jurdziński. Small progress measures for solving parity games. In *Proc. 17th Symp. Theoretical Aspects of Comp. Sci. (STACS)*, pages 290–301, 2000.
[KSBG02] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Explointing local similarity for indexing of paths in graph structured data. In *Proc. of 18th Int. Conf. on Data Engineering (ICDE)*, 2002.
[Saf88] S. Safra. On the complexity of ω-automata. In *Proc. of 29th Ann. Symp. on Foundations of Comp. Sci.*, pages 319–327, 1988.
[SB00] F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *Proceedings of 12th Int. Conf. on Computer Aided Verification*, 2000.
[VW86] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st Symp. on Logic in Comp. Sci. (LICS)*, pages 322–331, 1986.

A Proof of Theorem 3

One direction of the proof is easy: $L_\omega(A) \subseteq L_\omega(A/{\approx_k^{de}})$. Suppose $q_0a_0q_1a_1\ldots$ is an accepting run of A, then $[q_0]a_0[q_1]a_1\ldots$ constitutes an accepting run of $A/{\approx_k^{de}}$.

The other direction, $L_\omega(A/{\approx_k^{de}}) \subseteq L_\omega(A)$, will require some lemmas. Before going into the lemmas, we note that for any type of k-simulation, including delayed k-simulation, we need not consider k-tuples $\mathbf{q}_i$, but rather we can view $\mathbf{q}_i$ as sets of at most k states (call them "k-sets") together with a canonical mapping, to be described, from pebbles to those states. Assume the states $Q = \{q_1, \ldots, q_n\}$ are ordered 1 through n. Instead of viewing $\mathbf{q}_i$ as a k-tuple, the only thing that matters is the set of states $\{q_{i_1}, \ldots, q_{i_d}\}$ that are in the k-tuple $\mathbf{q}_i$. Assume $i_1 < i_2 < \ldots < i_d$. We can assume that pebbles are always mapped to states in the following canonical way: for pebbles $j \in \{1, \ldots, d\}$, $Blue_j$ gets mapped to q_{i_j}, and for $j \in \{d+1, \ldots, k\}$, $Blue_j$ gets mapped to q_{i_d}. Note that the transition relation $\Gamma_{A,k}$ remains identical on k-sets as on k-tuples, i.e., whether or not a transition $(\mathbf{q}_i, a, \mathbf{q}_j)$ is in $\Gamma_{A,k}$ is unaffected by whether we view $\mathbf{q}_i, \mathbf{q}_j$ as k-vectors or k-sets. Moreover, a k-vector $\mathbf{q}_j$ in a sequence $\pi = \mathbf{q}_0a_0\mathbf{q}_1\ldots$ will be good since some prior round i if and only if the corresponding k-set $\mathbf{q}_j$ is also good since round i, meaning that each pebble has seen an accept state since round i and j is the least such round $\geq i$.[4] Thus there is no loss of power on the part of Duplicator in viewing $\mathbf{q}_i$ as k-sets, and in fact we could have defined k-simulations in terms of k-sets, but didn't do so only for convenience in some other proofs. Henceforth, we assume $\mathbf{q}_i$ are k-sets.

Lemma 1. *On an automaton A with n states, the binary relation $\sqsubseteq_n^{de}$ is transitive. Hence $q \preceq_n^{de} q' \iff q \sqsubseteq_n^{de} q'$.*

Proof. Suppose, using the n-set notation, that $q \sqsubseteq_n^{de} \{q'\}$ and that $q' \sqsubseteq_n^{de} \{q''\}$. We will show that $q \sqsubseteq_n^{de} \{q''\}$.

Suppose Spoiler plays a run $\pi = q_0 a_0 q_1 a_1 \ldots$ in the game $\mathbf{G}_A^{de}(q, \mathbf{q}'_0)$, where $q_0 = q$ and $\mathbf{q}'_0 = \{q'\}$. Duplicator's winning strategy in that game supplies a sequence $\pi' = \mathbf{q}'_0 a_0 \mathbf{q}'_1 a_1 \ldots$, where each $\mathbf{q}'_i$ is an n-set.

We want to supply a sequence $\pi'' = \mathbf{q}''_0 a_0 \mathbf{q}''_1 a_1 \mathbf{q}''_2 \ldots$ such that $\mathbf{q}''_0 = \{q''\}$ and such that Duplicator wins the game $\mathbf{G}_A^{de}(q_0, \mathbf{q}''_0)$ when Spoiler and Duplicator play π and π'', respectively. In constructing π'', we can use Duplicator's winning strategy in the game $\mathbf{G}_A^{de}(q', \mathbf{q}''_0)$. Consider any run $\rho = q_0^\rho a_0 q_1^\rho a_1 \ldots$, such that $q_i^\rho \in \mathbf{q}'_i$. Duplicator, according to its winning strategy, can respond to this run with a winning sequence $\pi''_\rho = \mathbf{q}_0^\rho a_0 \mathbf{q}_1^\rho a_1 \ldots$. We use these sequences to construct Duplicator's response in the game $\mathbf{G}_A^{de}(q_0, \mathbf{q}''_0)$.

Duplicator's strategy is as follows: suppose that thus far Duplicator has constructed a response $\mathbf{q}''_0 a_0 \mathbf{q}''_1 \ldots \mathbf{q}''_i$ to $q_0 a_0 \ldots a_{i-1} q_i$. In response to the move to q_{i+1} via a_i, duplicator will respond with $\mathbf{q}''_{i+1} = \cup_\rho \mathbf{q}_{i+1}^\rho$, where the union is over all runs ρ with each $q_i^\rho \in \mathbf{q}'_i$. In other words, Duplicator's response will be to move to the <u>union</u> of all sets that are responses of Duplicator in its winning strategy against any run in the sequence π', which is itself the sequence that provides a win for Duplicator in the game $\mathbf{G}_A^{de}(q, \mathbf{q}')$.

Note that since there are at most n states, we need not worry that the size of the union is too large to be accommodated by n pebbles. We need only show that indeed π'' provides a win for Duplicator in response to π. For this we have to establish that if $q_i \in F$ then there is some $\mathbf{q}''_i$ which is good since round i. We use the fact that if $q_i \in F$ then for each π'_ρ, there exists a $j \geq i$ such that $\mathbf{q}_j^\rho$ is good since round i. The crucial observation is that since every pebble in $\mathbf{q}_j^\rho$ has seen an accept state since round i, then for every $j' \geq j$ also, every pebble in $\mathbf{q}_{j'}^\rho$ has seen an accept state since round i. Thus, we let $j^\star$ be the maximum, over all ρ, of the minimum $j \geq i$ such that $\mathbf{q}_j^\rho$ is good since round i, then by construction of π'', there is a j'', $i \leq j'' \leq j^\star$, such that $\mathbf{q}''_{j''}$ is good since round i. That establishes the Lemma. □

Lemma 2. *For a state $q_0 \in Q$ and an n-set $\mathbf{q}'_0$:*

[4] The order in which pebbles see accept states might change, because of the canonical mapping of pebbles to states, but this is irrelevant. Note that if two pebbles in a tuple $\mathbf{q}_j$ are on the same state, then they either both have seen an accept state since some prior round i or they both have not.

1. *If* $q_0 \sqsubseteq_n^{de} \mathbf{q}_0'$ *and* $(q_0, a, q_1) \in \Delta$, *then there is an* n*-set* $\mathbf{q}_1'$ *such that* $(\mathbf{q}_0', a, \mathbf{q}_1') \in \Gamma_A$, *and such that* $q_1 \sqsubseteq_n^{de} \mathbf{q}_1'$. *Moreover,* $\mathbf{q}_1'$ *can be taken to be Duplicator's response, according to its winning strategy in* $\mathbf{G}_A^{de}(q_0, \mathbf{q}_0')$, *to a move by spoiler using* (q_0, a, q_1).
2. *If* $q_0 \sqsubseteq_n^{de} \mathbf{q}_0'$ *and* $[q_0]a_0[q_1]a_1 \ldots$ *is a run of* $A/\approx_n^{de}$ *then there exists a sequence* $\pi' = \mathbf{q}_0' a_0 \mathbf{q}_1' a_1 \ldots$ *such that* $q_i \sqsubseteq_n^{de} \mathbf{q}_i'$ *and* $(\mathbf{q}_i', a_i, \mathbf{q}_{i+1}') \in \Gamma_A$. *Moreover, if* $q_i \in F$ *then for some* $r \geq i$, $\mathbf{q}_r'$ *is good since round* i.

Proof. For the first part, we know Duplicator has a winning strategy in $\mathbf{G}_A^{de}(q_0, \mathbf{q}_0')$. Let $\mathbf{q}_1'$ be Duplicator's response, according to its winning strategy, to a move by Spoiler using edge (q_0, a, q_1). Then $(\mathbf{q}_0', a, \mathbf{q}_1') \in \Gamma_A$. Now, it isn't hard to see that a winning strategy for Duplicator in $\mathbf{G}_A^{de}(q_1, \mathbf{q}_1')$ is given by playing according to Duplicator's winning strategy in $\mathbf{G}_A^{de}(q_0, \mathbf{q}_0')$ after the initial moves of *Red* to q_1 and *Blue* to $\mathbf{q}_1'$.

The second part can be established from the first by induction, using the transitivity of $\sqsubseteq_n^{de}$ established in Lemma 1. The proof is similar to the proof of Lemma 1, and we will not provide all the details here. In constructing the sequence π', we repeatedly combine duplicator strategies by taking the union of the n-sets in those strategies. Note that for all i, there are states $q_i^\star$ and $q_{i+1}^\star$ in $[q_i]$ and $[q_{i+1}]$, respectively, such that $(q_i^\star, a_i, q_{i+1}^\star) \in \Delta$. Thus, since $q_i^\star \sqsubseteq_n^{de} q_i$, and using the transitivity established in Lemma 1, there is a set $\mathbf{q}_{i+1}''$ such that $(\{q_i\}, a_i, \mathbf{q}_{i+1}'') \in \Gamma_A$, and such that $q_{i+1} \sqsubseteq_n^{de} \mathbf{q}_{i+1}''$. Having, by induction, already constructed $\mathbf{q}_i'$, we now construct $\mathbf{q}_{i+1}'$ as the union $\cup_\lambda \mathbf{q}^\lambda$ of the sets $\mathbf{q}^\lambda$ which would constitute a response by duplicator from $\mathbf{q}_i'$ to a move by spoiler from q_i to each $q_\lambda \in \mathbf{q}_{i+1}''$. □

To complete the proof of Theorem 3, we need to establish that if $\pi = [q_0]a_0[q_1]a_1 \ldots$ is an accepting run of $A/\approx_k^{de}$, then there is an accepting run π' of A over the same word w_π. Note that by part 1 of Theorem 1, delayed n-simulation is at least as large a relation as delayed k-simulation, and thus $\pi = [q_0]a_0[q_1]a_1 \ldots$ can also be viewed as an accepting run of $A/\approx_n^{de}$. Hence, using the sequence π' obtained in part 2 of Lemma 2, we can build an accepting run of A on w_π using the same construction as that used to prove part 3 of Theorem 2. This completes the proof.