Abstraction for Branching Time Properties

Kedar S. Namjoshi

Bell Labs, Lucent Technologies
kedar@research.bell-labs.com

Abstract. Effective program abstraction is needed to successfully apply model checking in practice. This paper studies the question of constructing abstractions that preserve branching time properties. The key challenge is to simultaneously preserve the existential and universal aspects of a property, *without* relying on bisimulation. To achieve this, our method abstracts an *alternating transition system* (ATS) formed by the product of a program with an alternating tree automaton for a property. The AND-OR distinction in the ATS is used to guide the abstraction, weakening the transition relation at AND states, and strengthening it at OR states. We show *semantic completeness*: i.e., whenever a program satisfies a property, this can be shown using a *finite-state* abstract ATS produced by the method. To achieve completeness, the method requires *choice predicates* that help resolve nondeterminism at OR states, and rank functions that help preserve progress properties. Specializing this result to predicate abstraction, we obtain exact characterizations of the types of properties provable with these methods.

1 Introduction

It is generally accepted that effective, automated, program abstraction is central to the successful application of model checking techniques. Methods include the use of transformations (e.g., [11, 21]), and predicate abstraction (e.g., [19, 4, 29, 3, 33, 20]). Most current methods use abstractions that preserve universal temporal properties, such as those expressible in linear temporal logic (LTL) and ACTL*. In several settings, there is a need for methods that preserve the full range of branching time properties, including mixed existential and universal ones – for instance, to analyze programs with unresolved non-determinism, or to analyze process-environment interaction [1].

The key challenge is to simultaneously preserve both the universal and the existential aspects of a branching time property during abstraction. Methods for universal properties rely on establishing a simulation relation [26] from the concrete to an abstract program. To preserve all branching time properties in a similar manner requires a bisimulation between the two programs, which is too restrictive. We propose a method that abstracts not the program, but rather the product *alternating transition system* (ATS) that is formed from the program and an alternating tree automaton for the property. The ATS can be abstracted through local transformations, weakening the transition relation at AND states, and strengthening it at OR states, in a manner similar to that of [14].

As the verification problem is undecidable in general even for invariance properties, no algorithm exists that can always construct a finite-state abstract program precise enough to prove that a concrete program satisfies a property. Hence, we look for *semantic completeness*: if a program satisfies a property, can this fact be shown using a finite state abstract program constructed by the method, ignoring computability issues? (In practice, the computability question corresponds to using a powerful enough decision procedure.) Uribe [32] and Kesten and Pnueli [22] showed that simulation-based abstraction must be augmented with fairness to be complete for LTL progress properties, such as termination. The fairness constraints serve to abstractly represent such termination requirements.

This completeness result does not, however, apply to all universal properties; for instance, to logics such as ACTL. Technically, the problem is that disjunction of temporal state formulas, as in $\mathsf{AX}(p) \vee \mathsf{AX}(\neg p)$, cannot be represented in LTL. Thus, for branching time properties, we have the distinction of AND vs. OR branching in addition to the invariance-progress distinction that was considered earlier. This distinction is important: we show that to achieve completeness, one requires *choice predicates* at OR states, which provide hints for the resolution of the OR nondeterminism, in addition to the rank predicates needed for abstracting liveness properties. As in [32, 22], our completeness result links the *deductive provability* of a property on a program — using a proof system designed in [27] — to the construction of a finite abstract ATS. The analysis reveals that abstraction without any augmentation can be used only to verify those properties that have proofs with *uniform* OR choice (a precise definition is given later), and *bounded* progress measures, showing why both types of augmentation are needed to verify arbitrary properties.

We examine the consequences of these results for *predicate abstraction*, which defines the abstract state space in terms of boolean variables that correspond to concrete program predicates. Automatically discovering relevant predicates is a key problem, for which several heuristics have been proposed (e.g., [29, 8]). Based on our completeness results, we can give an exact characterization of the properties provable using such discovery methods.

To summarize, the main contribution of this paper is a new abstraction method for branching time properties, which does not rely on bisimulation. This is the first such method for branching time properties that is known to be complete. As corollaries, we derive several completeness results for predicate discovery procedures. In addition, a key intermediate theorem shows how to construct a deductive proof of correctness on the concrete state space from a feasible abstract ATS. Such proof construction is of independent interest [28].

2 Preliminaries

In this section, we recall the definition of alternating tree automata (ATA), and the product construction used in model checking. For the rest of the paper, we fix an action set Σ and a set of atomic propositions, AP.

An *labeled transition system* (LTS) M over Σ and AP is is defined by a tuple (S, I, R, L), where S is a set of states, I is the subset of initial states, $R \subseteq S \times \Sigma \times S$ is a transition relation, and $L : S \to 2^{AP}$ is a labeling of states with atomic propositions. We assume that the relation R is total over Σ; i.e., for each a in Σ, every state has an a-successor.

An *alternating tree automaton* (ATA) over Σ and AP is given by a tuple $(Q \cup \{tt, ff\}, \hat{q}, \delta, F)$, where Q is a finite set of states, $\hat{q}$ in Q is the initial state, δ is a transition relation, and $F = (F_0, \ldots, F_{2n})$ is a partition of Q, called the *parity* acceptance condition [17]. A sequence of automaton states is accepted if *the least* index i such that a state from F_i occurs infinitely often on the sequence is *even*. The transition relation δ maps an automaton state, and a predicate on AP to one of: *ff* (an error state); *tt* (an accept state); $q_1 \wedge q_2$ (forking off automaton copies in state q_1 and q_2); $q_1 \vee q_2$ (choosing to proceed in either state q_1 or state q_2); $\langle a \rangle q_1$ (continuing in q_1 for some a-successor); $[a]q_1$ (continuing in q_1 for every a-successor).

An LTS $M = (S, I, R, L)$ *satisfies* a property given by an ATA $A = (Q, \hat{q}, \delta, F)$ iff player I has a winning strategy in an infinite, two player game [17]. In this game, a configuration is a pair of a computation tree node of M labeled by a state s, and an automaton state q. A configuration labeled (s, q) is a win for player I if $\delta(q, L(s)) = tt$; it is a loss if $\delta(q, L(s)) = ff$. For other values of δ, player I picks the next move iff $\delta(q, L(s))$ is either $\langle a \rangle q_1$ or $q_1 \vee q_2$. Player I picks an a-successor for s for $\langle a \rangle q_1$, or the choice of disjunct. Similarly, player II picks an a-successor for s for $[a]q_1$, or the choice of conjunct for $q_1 \wedge q_2$. A play of the game is a win for player I iff it either ends in a win for I, or it is infinite and the sequence of automaton states on it satisfies F. A *strategy* is a function mapping a partial play to the next move; given strategies for players I and II, one can generate the possible plays. Finally, the LTS satisfies the automaton property iff player I has a *winning* strategy (one for which every generated play is a win for player I) for the game played on the computation tree of the LTS from the initial configuration labeled $(\hat{s}, \hat{q})$.

Every closed formula of the μ-calculus [24] can be translated in linear time to an equivalent ATA [17]. The transition relation of the automaton has the following simple form: each state has a single transition for the input predicate *true*, except for a transition to *tt* on predicate l; in which case, there is another transition to *ff* on $\neg l$. In the rest of the paper, we work with such simple automata.

An *alternating transition system* (ATS) over a set Γ of state labels is defined by a tuple (S, I, R, L), where S, the set of states, is partitioned into AND and OR subsets, I is a set of initial states, $R \subseteq S \times S$ is the transition relation, and $L : S \to \Gamma$ is the state labeling.

The *model checking* problem is to determine whether M satisfies A at all initial states, and is written as $M \models A$. This can be determined using a product ATS, $M \times A$, defined by (S, I, R, L), where $S = S_M \times Q_A$, $I = I_M \times \{\hat{q}_A\}$, and $L : (s,q) \to q$. $R((s,q),(s',q'))$ holds based on the value of $\delta_A(q,l)$, as follows: (i) ff, tt: $q' = \delta_A(q,l) \wedge s' = s \wedge l(s)$, (ii) $q_1 \wedge q_2, q_1 \vee q_2$: $q' \in \{q_1, q_2\} \wedge s' = s$, as $l \equiv \mathit{true}$, and (iii) $\langle a \rangle q_1, [a]q_1$: $q' = q_1 \wedge R_M(s,a,s')$, as $l \equiv \mathit{true}$. A state is an OR state if $\delta_A(q,l)$ is either $q_1 \vee q_2$ or $\langle a \rangle q_1$, and an AND state otherwise. In [16], it is shown that M satisfies A iff player I has a winning strategy in the game graph defined by $M \times A$, where player I makes choices at OR states, and player II at AND states, and that this can be determined by model checking.

3 The Abstraction Method

We define our abstraction method for the product alternating transition system (ATS). Soundness is shown by constructing a valid concrete proof of correctness given the feasibility of the abstract ATS. This construction also indicates why augmentation with choice predicates and rank functions is needed in general. In the following, let $M = (S, I, R, L)$ be an LTS over Σ and AP, and let $A = (Q, \hat{q}, \delta, F)$, where $F = (F_0, \ldots, F_{2n})$ (for some n), be an ATA over Σ and AP. Let $M \times A$ be the product ATS, constructed as shown earlier.

The basic abstraction idea is simple: we are given an abstract domain $\overline{S}$, and a set of left-total *abstraction relations* $\{\xi_q \mid q \in Q\}$, where each $\xi_q \subseteq S \times \overline{S}$. We also define ξ_{tt} and ξ_{ff} to be $S \times \overline{S}$, and say that (s,q) is related to (t,q) iff $s \xi_q t$ holds. We abstract the ATS $M \times A$ by *weakening* its transition relation at AND states (thus allowing "more" transitions), and *strengthening* it at OR states (thus "eliminating" some transitions). The result is an abstract ATS, denoted by $\overline{M \times A}$. The abstract ATS is given by (S', I', R', L').

- The abstract set of states, $S' = \overline{S} \times Q$. The abstract OR states are those where $\delta(q, \mathit{true})$ has the form $q_1 \vee q_2$ or $\langle a \rangle q_1$, all others are AND states. Thus, related concrete and abstract states have the same AND/OR tag.
- The abstract state $(t, \hat{q})$ is in I' if there exists $s \in I$ for which $s \xi_{\hat{q}} t$.
- The abstract transition relation, R', is given by:
  - For an abstract AND state (t,q), the transition $((t,q),(t',q'))$ is in R' *if* there exists a concrete state (s,q) and a successor (s',q') that are related to $(t,q),(t',q')$ respectively.

$$R'((t,q),(t',q')) \;\Leftarrow\; (\exists s : s \xi_q t : (\exists s' : s' \xi_{q'} t' : R((s,q),(s',q'))))$$

  - For an abstract OR state (t,q), the transition $((t,q),(t',q'))$ is in R' *only if* for every (s,q) which is related to (t,q), there exists a successor (s',q') which is related to (t',q').

$$R'((t,q),(t',q')) \;\Rightarrow\; (\forall s : s \xi_q t : (\exists s' : s' \xi_{q'} t' : R((s,q),(s',q'))))$$

- The abstract labeling function L' maps a state (t,q) to q.

Consistency: For each $q \in Q$, $[\phi_q \Rightarrow (\exists k : (\rho_q = k))]$ (ρ_q is defined for every state in ϕ_q)
Initiality: $[I \Rightarrow \phi_{\hat{q}}]$ (every initial state satisfies the initial invariant)
Invariance and Progress: For each $q \in Q$, and predicate l over AP, based on the form of $\delta(q, l)$, check the following.

- tt: there is nothing to check.
- $f\!f$: $[\phi_q \Rightarrow \neg l]$ holds,
- $q_1 \wedge q_2$: $[\phi_q \wedge l \wedge (\rho_q = k) \Rightarrow (\phi_{q_1} \wedge (\rho_{q_1} \lhd_q k)) \wedge (\phi_{q_2} \wedge (\rho_{q_2} \lhd_q k))]$
- $q_1 \vee q_2$: $[\phi_q \wedge l \wedge (\rho_q = k) \Rightarrow (\phi_{q_1} \wedge (\rho_{q_1} \lhd_q k)) \vee (\phi_{q_2} \wedge (\rho_{q_2} \lhd_q k))]$
- $\langle a \rangle q_1$: $[\phi_q \wedge l \wedge (\rho_q = k) \Rightarrow \langle a \rangle (\phi_{q_1} \wedge (\rho_{q_1} \lhd_q k))]$
- $[a] q_1$: $[\phi_q \wedge l \wedge (\rho_q = k) \Rightarrow [a](\phi_{q_1} \wedge (\rho_{q_1} \lhd_q k))]$

Fig. 1. Deductive Proof System for Automaton Properties

Precise Abstraction: Notice that the method allows some flexibility in the definition of R'. If R' is defined in a way that the implications become equivalences, we say that R' is *precise* (precision of abstractions is studied in depth in [10, 14]). This flexibility can be exploited in practice by doing approximate but faster calculations of a less precise R'. Precise abstractions are needed for completeness, though, as is shown later. Note also that the abstract ATS can be constructed by symbolic calculations, thus avoiding the explicit construction of $M \times A$.

Theorem 0 ***(Soundness)*** *For any LTS M and alternating tree automaton A, let $\overline{M \times A}$ be defined by the abstraction method, based on a set of abstraction relations $\{\xi_q\}$. Then, if $\overline{M \times A}$ is feasible, so is $M \times A$.*

One can prove this theorem by showing that the relation α given by: $(s, q)\alpha(t, q')$ iff $q = q'$ and $s \xi_q t$ is an alternating refinement relation [1] which preserves the labeling (i.e., the automaton component). Therefore, any winning strategy for player I in $\overline{M \times A}$, induces (through the refinement) a winning strategy on $M \times A$. However, we use a different argument, showing how to construct a deductive proof that M satisfies A, given that $\overline{M \times A}$ is feasible. This construction provides information useful for the completeness proof.

Deductive Proofs: A deductive proof system (from [27]) for proving that M satisfies A is shown in Fig. 1. One needs: (i) for each automaton state q, an *invariance* predicate, ϕ_q, which is a subset of S, (ii) non-empty, well ordered sets $W_1, \ldots, W_n$ with associated partial orders $\preceq_1, \ldots, \preceq_n$. Let $W = W_1 \times \ldots \times W_n$, and let $\preceq$ be the lexicographic well order defined on W from $\{\preceq_i\}$, (iii) for each automaton state q, a partial *rank function* $\rho_q : S \to W$. Let $\prec^i$ be the restriction of $\prec$ to the first i components. For an automaton state q, the rank change predicate $(a \lhd_q b)$ holds either if q belongs to an odd indexed F_{2i-1} and $a \prec^i b$, or if q is in an even indexed F_{2i} and $a \preceq^i b$ (this odd/even distinction is clearly related to the parity condition). A proof is valid if it meets the conditions given the figure.

Theorem 1 *[27] For program M and automaton A, $M \models A$ iff there is a valid deductive proof that M satisfies A.*

Theorem 2 ***(Proof Construction)*** *If $\overline{M \times A}$ is feasible, there is a valid deductive proof that M satisfies A.*

Proof Sketch. Let $\mathcal{A}$ be the set of states that are wins for player I in $\overline{M \times A}$. The proof construction is similar to that in [27] so, for lack of space, we present only a short sketch. The proof Π is defined by (ϕ, ρ, W), where: (i) $\phi_q(s)$ holds iff there exists t such that $s\xi_q t$ and (t, q) is in $\mathcal{A}$, (ii) $\rho_q(s)$ is the minimum of the *signatures* of states t such that $s\xi_q t$ and $(t, q) \in \mathcal{A}$, and (iii) W is the domain of the signatures. The signature [31] of a state in $\mathcal{A}$ is an n-tuple of ordinals which, roughly, measures the progress made toward satisfying A's acceptance condition. For example, if A has a Büchi acceptance condition, it is the distance in $M \times A$ to an accepting state.

We consider in detail only the case where $\delta(q, \mathit{true}) = \langle a \rangle q_1$. If $\phi_q(s)$ and $(\rho_q(s) = k)$ holds for any state s of M and any $k \in W$, from the definitions, there is t such that $s\xi_q t$, $(t, q) \in \mathcal{A}$, and t has signature k. As (t, q) is an OR state, it has a successor, (t', q_1), in $\mathcal{A}$. By the $\forall\exists$ abstraction for OR states, (s, q) must have a successor (s', q_1) such that $s'\xi_{q_1} t'$. Thus, s' is an a-successor of s in M, and $\phi_{q_1}(s')$ holds. By a property of signatures [31], the signature of (t', q_1) is k', where $k' \lhd_q k$. By definition, $\rho_{q_1}(s') \preceq k'$, so that $\rho_{q_1}(s') \lhd_q k$. $\square$

Proof of Theorem 0: The soundness theorem follows from the combination of Theorems 1 and 2. If $\overline{M \times A}$ is feasible, by Theorem 2, there is a valid concrete proof that M satisfies A. Applying Theorem 1, it follows that M satisfies A. $\square$

Theorem 2 gives valuable information about the constructed proof Π if $\overline{M \times A}$ is finite-state:

- **(Uniform OR Choice)** By the $\forall\exists$ nature of the abstraction at OR states, for q such that $\delta(q, \mathit{true}) = q_1 \vee q_2$, if (t, q) has a transition to (t', q_1), then for every s such that $s\xi_q t$ holds, there is a transition from (s, q) to (s, q_1). A similar observation holds for the $\langle a \rangle$ case.
- **(Bounded Progress)** As the abstract ATS is finite-state, the rank domain W is a finite set. Thus, Π shows that M satisfies A using only bounded progress measures.

These restrictions mean that the abstraction procedure, although sound, cannot be complete. To illustrate this, consider showing the property $\mathsf{EF}(x \geq 0)$ (i.e., there exists a future where $x \geq 0$) for the following program M.

```
var x: integer; initially true
actions   (a) x := x-1  (b) x := x+1
```

The property is true at every initial state, but showing this requires a proof with unbounded progress measure (the measure $\rho(x) = -x,\ \mathit{if}\ x < 0,\ \mathit{else}\ 0.$). The automaton A for the property is defined below, and a fragment of the product ATS is shown in Fig. 2.

States: $\{q_0, q_1, q_2, q_3, q_4\}$; Initial state: q_0
Transitions: $\delta(q_0, \mathit{true}) = q_1 \ \vee\ q_2$; $\delta(q_1, x \geq 0) = \mathit{tt}$; $\delta(q_1, x < 0) = \mathit{ff}$;
$\delta(q_2, \mathit{true}) = q_3 \ \vee\ q_4$; $\delta(q_3, \mathit{true}) = \langle a \rangle q_0$; $\delta(q_4, \mathit{true}) = \langle b \rangle q_0$
Parity condition: $(\{q_1\}, \{q_0, q_2, q_3, q_4\})$

$$\begin{tikzcd}
(-2,q_0) & & & (-1,q_0) \arrow[dl] \arrow[dr] & & & & & & (0,q_0) \arrow[dl] \arrow[dr] & & & & & (1,q_0) \\
& & (-1,q_1) \arrow[dl] & & (-1,q_2) \arrow[dl] \arrow[dr] & & & & (0,q_1) \arrow[dl] & & (0,q_2) \arrow[dl] \arrow[dr] & & & & \\
& (-1,\mathit{ff}) & & (-1,q_3) \arrow[uulll] & & (-1,q_4) \arrow[uurrrr] & & (0,\mathit{tt}) & & (0,q_3) \arrow[uullllll] & & (0,q_4) \arrow[uurrr] & & &
\end{tikzcd}$$

Fig. 2. A portion of $M \times A$

Now consider the abstract ATS in Fig. 3. Here, *neg* stands for $\{x \,|\, (x < 0)\}$ and *nneg* for $\{x \,|\, (x \geq 0)\}$. Solid lines indicate transitions that are in the precise abstraction. However, these transitions, in themselves, are not sufficient for feasibility since there is no way to pick transitions so that it is possible to satisfy the acceptance condition from the abstract state (neg, q_0). We would like, in particular, the dashed transitions from state (neg, q_4) to exist in the abstraction, but these are ruled out by the strong $\forall\exists$ nature of the abstraction at OR states. Clearly, it is not possible for all states where $x < 0$ to have a b-transition to a state where $x < 0$, and similarly, it is not possible for all such states to have a b-transition to a state where $x \geq 0$. On the other hand, the $\forall\exists$ abstraction is needed for soundness at OR states. Moreover, no finite refinement of the *neg* state will resolve this problem.

$$\begin{tikzcd}
& & (\mathit{neg},q_0) \arrow[dl] \arrow[dr] & & & & & (\mathit{nneg},q_0) \arrow[dl] \arrow[dr] & & \\
& (\mathit{neg},q_1) \arrow[dl] & & (\mathit{neg},q_2) \arrow[dl] \arrow[dr] & & & (\mathit{nneg},q_1) \arrow[dl] & & (\mathit{nneg},q_2) \arrow[dl] \arrow[dr] & \\
(\mathit{neg},\mathit{ff}) & & (\mathit{neg},q_3) \arrow[uu] & & (\mathit{neg},q_4) \arrow[uull, dashed, bend right] \arrow[uurrr, dashed] & (\mathit{nneg},\mathit{tt}) & & (\mathit{nneg},q_3) & & (\mathit{nneg},q_4) \arrow[uull, bend right]
\end{tikzcd}$$

Fig. 3. A Possible Abstraction

Choice Predicates: A way out of this dilemma is given by the introduction of *choice predicates.* The essential idea is to weaken the $\forall$ quantification at an abstract OR state (t, q) in the $\forall\exists$ abstraction to apply only to a subset of the states in $\xi_q^{-1}(t)$. These subsets are supplied to the abstraction procedure through a partial function ϵ, defined for a subset of OR-states, called the *choice states* in the sequel. At each choice state (t, q), the function supplies, for a possible transition to a state (t', q'), a predicate $\epsilon((t,q),(t',q'))$ (note that the transition is possible if either $\delta(q, \mathit{true}) = q_1 \ \vee \ q_2$, and $q' \in \{q_1, q_2\}$, or $\delta(q, \mathit{true}) = \langle a \rangle q_1$ and $q' = q_1$). An abstract transition $((t,q),(t',q'))$ from a choice state (t,q) is computed by restricting the $\forall$ quantification in the $\forall\exists$ abstraction to states satisfying its choice predicate. The union of choice predicates for all transitions in R' from (t,q) should be a superset of $\xi_q^{-1}(t)$.

At (neg, q_4), we let the choice predicate for the transition to (neg, q_0) be $(x < -1)$, and the predicate for the transition to (nneg, q_0) be $(x = -1)$. This adds back the dashed transitions in Fig. 3. However, it also creates a different problem. The transition from (neg, q_4) to (neg, q_0) introduces an infinite loop, which does not exist in the original ATS, since this transition increments the value of x. To solve this difficulty, we use rank functions in a manner similar to that in [32, 22].

We suppose that a set of rank function $\{\eta_q \mid q \in Q\}$ is supplied, which map states in S to a well-founded set with the structure specified in the proof system. We add to each abstract transition a label ('good' or 'bad') indicating whether the rank given by η changes in a way appropriate to the change of automaton state along the transition. The final method, for the supplied abstraction relations $\{\xi_q\}$, rank functions $\{\eta_q\}$, and the choice function ϵ is given below. The abstract ATS is given by (S', I', R', L'), where S', I', and L' are defined as before. The definition of the abstract transition relation, R', is modified to the following, where g is the label on the abstract transition.

– For an abstract AND state (t,q),

$$R'((t,q),g,(t',q')) \ \Leftarrow \\ (\exists s : s\xi_q t : (\exists s' : s'\xi_{q'} t' : R((s,q),(s',q')) \ \wedge \ g \ \equiv \ \eta_{q'}(s') \lhd_q \eta_q(s)))$$

– For an abstract choice state (t,q),

$$R'((t,q),g,(t',q')) \Rightarrow \epsilon((t,q),(t',q')) \neq \emptyset \wedge (\forall s : s\xi_q t \ \wedge \ s \in \epsilon((t,q),(t',q')) : \\ (\exists s' : s'\xi_{q'} t' : R((s,q),(s',q')) \ \wedge \ g \ \equiv \ \eta_{q'}(s') \lhd_q \eta_q(s)))$$

– The transitions from abstract OR states are as for choice states with choice predicate $\epsilon \ \equiv \ \mathit{true}$.

Taking, in addition to the choice augmentation discussed above, the rank functions where for s such that $x(s) < 0$, $\eta_{q_0}(s) = -3 * x(s), \eta_{q_1}(s) = -3 * x(s) - 1, \eta_{q_2} = -3 * x(s) - 2$, and for $x(s) \geq 0$, all values are 0, and applying the augmented abstraction procedure, we obtain the abstract ATS shown in Fig. 4, where η-good transitions are labeled with $*$.

We only consider abstract ATS's where at a choice state (t,q), the union of the choice predicates on its successors together form a superset of $\xi_q^{-1}(t)$. We define a game on such abstract ATS which is identical to the game defined in Section 2, except that: (a) player II has the choice of successor at every choice state, and (b) all infinite plays satisfy *either* the parity acceptance condition of the

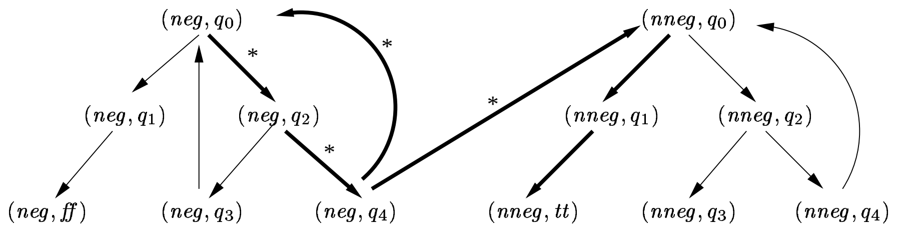

Fig. 4. Abstract ATS after Augmented Abstraction

automaton or, from some point on, contain only η-good transitions. We say that an abstract ATS is *subtly feasible* if player I has a winning strategy in the new game. The bold transitions in Fig. 4 indicate such a winning strategy.

Theorem 3 ***(Soundness of Augmented Abstraction)*** *For any LTS M and alternating tree automaton A, let $\overline{M \times A}$ be defined by the augmented abstraction procedure above, based on a set of abstraction relations $\{\xi_q\}$, choice predicate ϵ, and rank functions $\{\eta_q\}$. If $\overline{M \times A}$ is subtly feasible, then $M \times A$ is feasible.*

Proof. As $\overline{M \times A}$ is subtly feasible, player I has a winning strategy on it. Let $\mathcal{A}$ be the set of winning states for player I. We use this strategy to provide a winning strategy for the player on $M \times A$. Inductively, we assume that for a state (s, q) on a play following this strategy, there is an abstract state (t, q) in $\mathcal{A}$ such that $s\xi_q t$ holds. This is true for the initial states of $M \times A$ as every initial state of the abstract ATS is in $\mathcal{A}$.

Consider any state (s, q) on a play, and let (t, q) be its corresponding abstract state. If (s, q) is an AND-state, let (s', q') be any successor chosen by player II. By the definition of the abstract ATS, there is a successor (t', q') of (t, q) that matches it. Now suppose that (s, q) is an OR state, and that (t, q) is a choice state. Let (t', q') be a successor of (t, q) (which is in $\mathcal{A}$ by the new game rules) such that $s \in \epsilon((t, q), (t', q'))$. Such a successor must exist because the choice predicates at (t, q) cover $\xi_q^{-1}(t)$. The $\forall\exists$ abstraction implies that (s, q) has a successor (s', q') such that $s'\xi_{q'}t'$ holds. Player I picks this successor. A similar argument applies to ordinary OR states.

Any constructed play π has a corresponding play π' in the winning game on the abstract ATS, which agrees with it on the sequence of automaton states. So if π is maximal and finite, it must end in a state with label *tt*. If π is infinite, we show by contradiction that it must satisfy F. If not, then π' does not do so either, and the new acceptance condition implies that it eventually consists of only η-good transitions. Thus, from some point on, the least F-index in π is *odd* (say $2k - 1$), and all of its transitions correspond to η-good transitions in π'. At each such concrete transition, the η-goodness condition, with the definition of $\lhd$, ensure that η, restricted to the first k components, does not increase, and strictly decreases infinitely often. This contradicts the well-foundedness of the rank domain. □

4 Completeness

Theorem 4 ***(Completeness)*** *If M satisfies an ATA property A, there is an augmented abstraction $\overline{M \times A}$ that is subtly feasible.*

Proof. Since M satisfies A, by Theorem 1, there is a valid proof $\Pi = (\phi, \rho, W)$ of this fact. The constructed abstraction follows the proof very closely, as is also the case for the constructions in [32, 22] for LTL.

Let the abstract domain $\overline{S}$ be the set $\{a_q \mid q \in Q\} \cup \{a_{tt}, a_{ff}\} \cup \{\bot\}$. Let the abstraction relations for $q \in Q$ be: $s \xi_q t$ iff $\phi_q(s) \wedge t = a_q$ or $\neg\phi_q(s) \wedge t = \bot$ holds. The state $\bot$ is an unreachable state used to ensure that each ξ_q is left-total. The rank functions used for abstraction are $\{\rho_q\}$ from the proof. The choice predicate is defined only for those abstract OR states (a_q, q) where $\delta(q, \mathit{true}) = q_1 \vee q_2$: $\epsilon((a_q, q), (a_{q'}, q'))$ is the set $\{s \mid \phi_{q'}(s) \wedge \rho_{q'}(s) \lhd_q \rho_q(s)\}$, and is empty for other possible successor states. We claim that the *precise* abstract program constructed with these choices is subtly feasible. Inductively, the winning strategy ensures that the only reachable configurations are those of the form (a_q, q), where q is in $Q \cup \{tt\}$, and ϕ_q is non-empty. This is true of the only initial state, $(a_{\hat{q}}, \hat{q})$.

Suppose (a_q, q) is reached according to some partial play. Since ϕ_q is non-empty, there is a related concrete state (s, q). If $\delta(q, \mathit{true}) = q_1 \wedge q_2$, then by the proof, s satisfies ϕ_{q_1} and ϕ_{q_2}. Thus, (a_q, q) has (a_{q_1}, q_1) and (a_{q_2}, q_2) as successor states. As the abstraction is precise, these are the only possible successors. Now suppose that $\delta(q, \mathit{true}) = [a]q'$. Then, by the proof, for every a-successor (s', q') of (s, q) in $M \times A$, $\phi_{q'}(s')$ holds, and there is at least one such successor, so that (a_q, q) has $(a_{q'}, q')$ as its only successor. If $\delta(q, l) = \mathit{ff}$ (so that $\delta(q, \neg l) = \mathit{tt}$), by the proof, $[\phi_q \Rightarrow \neg l]$, so (a_q, q) can only have the successor (a_{tt}, tt). Suppose that (a_q, q) is a choice state, so that $\delta(q, \mathit{true}) = q_1 \vee q_2$. By the proof, $\xi_q^{-1}(a_q) = \phi_q$ is covered by the two choices, so that every state in ϕ_q satisfies some choice, and that (a_{q_1}, q_1) and (a_{q_2}, q_2) are the only possible successors of this abstract state. Otherwise, (a_q, q) is an OR state and $\delta(q, \mathit{true}) = \langle a \rangle q'$. By the proof, every state s satisfying ϕ_q has an a-successor satisfying $\phi_{q'}$. Hence, (a_q, q) has a transition to $(a_{q'}, q')$, which is picked by player I.

It follows from the proof that every abstract transition is ρ-good. Thus, every infinite play either satisfies the parity acceptance condition or has only ρ-good transitions, and every maximal finite path in $\mathcal{A}$ must end with automaton component tt. Hence, the abstract ATS is subtly feasible. □

The abstract ATS constructed in this manner for the example program is given in Fig. 5, with the winning strategy outlined in bold (*int* stands for the set of all integers). This uses the same rank functions as for Fig. 4, and invariants defined by the abstract state component.

4.1 Predicate Abstraction

Predicate abstraction[19] defines the abstract state space in terms of boolean variables corresponding to concrete program predicates. Computing a relevant

$$\begin{tikzcd}
 & & (int, q_0) \arrow[dl, "*"'] \arrow[dr, "*"] & & \\
 & (nneg, q_1) \arrow[dl, "*"'] & & (neg, q_2) \arrow[dr, "*"] & \\
(nneg, tt) & & (\emptyset, q_3) \arrow[uu] & & (neg, q_4) \arrow[uull, bend right=60, "*"']
\end{tikzcd}$$

Fig. 5. Abstract ATS derived from the completeness proof

set of predicates is impossible in general [20]; however, there are several heuristics for *predicate discovery.* The general predicate discovery scheme [29, 8] starts with a set $\mathcal{P}_0$ of predicates from the correctness property, and iteratively computes $\mathcal{P}_{i+1}$ by adding the predicates in the weakest precondition *wp* [15] of $\mathcal{P}_i$ to $\mathcal{P}_i$. Let the limit of this procedure be denoted by the set $\mathcal{P}^*$.

From a modification of the main completeness theorem, we can derive the reverse direction of the following theorem. Note that the forward direction holds from the first soundness theorem. Thus, we obtain an exact characterization of the completeness of predicate discovery methods.

Theorem 5 ***(Relative Completeness of Predicate Discovery)*** *For an LTS M and ATA A, predicate discovery coupled with abstraction produces a feasible result iff there is a proof that M satisfies A where the invariants in the proof are constructed from predicates in $\mathcal{P}^*$, the progress ranks are bounded, and the OR choices are uniform.*

In [2], the authors show completeness of predicate discovery for invariance properties (i.e., $\mathsf{AG}(p)$) relative to an oracle which can widen intermediate results of the fixpoint computation of $\Phi = \mathsf{EF}(\neg p)$, by dropping some conjunctive terms. Notice that the widened fixpoint, Φ^+, is defined in terms of predicates from $\mathcal{P}^*$, and $\neg(\Phi^+)$ is an inductive invariant implying $\mathsf{AG}(p)$. Thus, a generalization of their result including existential and progress properties can be derived from these observations and Theorem 5. This method of proof also shows that the result holds for more powerful "clairvoyant" oracles, which may perform widening using predicates in $\mathcal{P}^*$ that do not appear at the current stage.

Theorem 6 ***(Bisimulation and Finite-state Completeness)*** *If LTS M has a finite bisimulation quotient preserving the predicates, AP, in a property A, then predicate discovery coupled with abstraction produces a feasible result.*

Proof Sketch. The proof hinges on the fact that symbolic algorithms for bisimulation minimization (e.g., [5, 25]) compute a finite quotient that is based on $\mathcal{P}^*$ for a suitable initial choice of $\mathcal{P}_0$. Bisimulation ensures that the conditions in Theorem 5 on progress ranks and OR choice are met. Consequently, predicate discovery always produces a feasible abstraction for finite-state systems. □

5 Related Work

There is a large literature on the links between program abstraction and model checking – we discuss here only the most closely related results. As mentioned earlier, [32, 22, 23] show that simulation augmented with progress hints is complete for linear time properties, based on earlier work on deductive approaches [6, 30]. Our abstraction method builds on this work, but uses progress hints differently, and handles branching time logics (such as the μ-calculus), which are more powerful than LTL and include both universal and existential modalities. We also offer a finer analysis of completeness, with and without progress hints.

Another closely related line of research is the work in [10, 13, 14] on abstraction for the μ-calculus and sub-logics, based on the abstract interpretation paradigm [12]. The abstract program has *two* transition relations, one used for checking existential properties, the other for universal ones. Our method uses similar $\exists\exists$ and $\forall\exists$ abstractions, but the duality is expressed locally by the alternation within the ATS. These papers study the precision of abstract interpretations, but not completeness. It is not known whether their abstraction methods are complete. From the results of this paper, however, this seems unlikely, since completeness seems to require both choice predicates and rank functions, applied in a manner closely tied to the property automaton. A completeness result is given in [9] for CTL*, but only under a strict congruence assumption on the concrete system. Partial transition systems (e.g., those with may and must relations) have been used to define abstractions preserving branching time properties [7, 18] – but with a "gap" where preservation is uncertain. Partial information is an orthogonal issue, and such methods can be incorporated in the analysis of ATS's. We have related our completeness results to those in [2] in the previous section.

Acknowledgements: Thanks go to Dennis Dams for very helpful discussions in the course of this work, and to the referees for several insightful comments.

References

1. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *COMPOS*, volume 1536 of *LNCS*, 1997. Journal version in JACM.
2. T. Ball, A. Podelski, and S. K. Rajamani. Relative completeness of abstraction refinement for software model checking. In *TACAS*, number 2280 in LNCS, 2002.
3. T. Ball and S. K. Rajamani. The SLAM toolkit. In *CAV*, volume 2102 of *LNCS*, 2001.
4. S. Bensalem, Y. Lakhnech, and S. Owre. InVeST: A tool for the verification of invariants. In *CAV*, volume 1427 of *LNCS*, 1998.
5. A. Bouajjani, J-C. Fernandez, and N. Halbwachs. Minimal model generation. In *CAV*, volume 531 of *LNCS*, 1990.
6. I.A. Browne, Z. Manna, and H.B. Sipma. Generalized temporal verification diagrams. In *FST&TCS*, volume 1026 of *LNCS*, 1995.
7. G. Bruns and P. Godefroid. Generalized model checking: Reasoning about partial state spaces. In *CONCUR*, volume 1877 of *LNCS*, 2000.

8. E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In *CAV*, volume 1855 of *LNCS*, 2000.
9. E.M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *TOPLAS*, 16(5), 1994.
10. R. Cleaveland, S. P. Iyer, and D. Yankelevich. Optimality in abstractions of model checking. In *SAS*, volume 983 of *LNCS*, 1995.
11. J. Corbett, M. Dwyer, J. Hatcliff, C. Pasareanu, Robby, S. Laubach, and H.Zheng. Bandera: extracting finite-state models from Java source code. In *ICSE*, 2001.
12. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, 1997.
13. D. Dams. *Abstract Interpretation and Partition Refinement for Model Checking.* PhD thesis, Eindhoven University of Technology, 1996.
14. D. Dams, O. Grumberg, and R. Gerth. Abstract interpretation of reactive systems. *TOPLAS*, 19(2), 1997.
15. E.W. Dijkstra. Guarded commands, nondeterminacy, and formal derivation of programs. *CACM*, 18(8), 1975.
16. E. A. Emerson, C.S. Jutla, and A.P. Sistla. On model-checking for fragments of μ-calculus. In *CAV*, 1993.
17. E.A. Emerson and C.S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FOCS*, 1991.
18. P. Godefroid, M. Huth, and R. Jagadeesan. Abstraction-based model checking using modal transition systems. In *CONCUR*, volume 2154 of *LNCS*, 2001.
19. S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In *CAV*, volume 1254 of *LNCS*, 1997.
20. T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Lazy abstraction. In *POPL*, 2002.
21. G. J. Holzmann and M. H. Smith. Automating software feature verification. *Bell Labs Technical Journal*, 5(2), 2000.
22. Y. Kesten and A. Pnueli. Verification by augmented finitary abstraction. *Information and Computation*, 163(1), 2000.
23. Y. Kesten, A. Pnueli, and M. Vardi. Verification by augmented abstraction: The automata-theoretic view. *JCSS*, 62(4), 2001.
24. D. Kozen. Results on the propositional mu-calculus. In *ICALP*, 1982.
25. D. Lee and M. Yannakakis. Online minimization of transition systems. In *STOC*, 1992.
26. R. Milner. An algebraic definition of simulation between programs. In *2nd IJCAI*, 1971.
27. K. S. Namjoshi. Certifying model checkers. In *CAV*, number 2102 in LNCS, 2001.
28. K. S. Namjoshi. Lifting temporal proofs across abstractions. In *VMCAI*, volume 2575 of *LNCS*, 2003.
29. K. S. Namjoshi and R. P. Kurshan. Syntactic program transformations for automatic abstraction. In *CAV*, volume 1855 of *LNCS*, 2000.
30. H.B. Sipma, T.E. Uribe, and Z. Manna. Deductive model checking. *Formal Methods in System Design*, 15(1), 1999.
31. R.S. Streett and E.A. Emerson. The propositional mu-calculus is elementary. In *ICALP*, 1984. Full version in *Inf. and Comp.* 81(3), pp. 249-264, 1989.
32. T.E. Uribe. *Abstraction-Based Deductive-Algorithmic Verification of Reactive Systems.* PhD thesis, Stanford University, 1999.
33. W. Visser, K. Havelund, G. Brat, and S. Park. Model checking programs. In *ICSE*, 2000.