

Ranked Predicate Abstraction for Branching Time: Complete, Incremental, and Precise*

Harald Fecher¹ and Michael Huth²

¹ Institut für Informatik, Christian-Albrechts-Universität zu Kiel, Germany
hf@informatik.uni-kiel.de

² Department of Computing, Imperial College London, United Kingdom
M.Huth@doc.imperial.ac.uk

Abstract. Predicate abstraction frameworks are a powerful means of combating the state explosion problem in model checking as they automatically synthesize abstract models that either verify compliance with a property, give rise to a genuine counter-example or produce a spurious counter-example that drives refinement of the abstract model. Prominent tools for safety (e.g. Blast) and termination (e.g. Terminator) checking rely on this approach. This paper presents such an abstraction framework for all properties of the modal μ -calculus based on ranked predicate abstraction. We show that our framework is incremental and confluent and should therefore allow good refinement heuristics. Moreover, ranked predicate abstractions are proved to be precise (i.e. optimal as abstractions) and also complete in that all properties true in a model are also true in a finite-state, ranked predicate abstraction of that model. This completeness relates to known characterizations of relative completeness for predicate abstraction with branching time.

1 Introduction

Model checking, invented 25 years ago [4, 24], provides a framework for verifying properties of systems: a system is represented by a mathematical model M , a property of interest is coded within a formal language as some ϕ , and the satisfaction relationship is captured by a formal predicate \models relating formal models and properties. Its instances $M \models \phi$ are then decided fully automatic (e.g. if M is finite-state) or semi-automatic (e.g. if M is abstracted first). Models often have infinite state space and this infinity can have a variety of sources: unbounded data-types, recursive process specifications, quantitative and continuous parameter values etc. Even if models are finite, their size is typically exponential in the number of system variables or communicating sub-models. This *state explosion problem* is a severe impediment to the scalability of this approach and its technology transfer into industrial research & development units even if the complexity of computing $M \models \phi$ is linear in the sizes of M and ϕ .

Abstraction of models, e.g. [20, 5, 7, 8, 3, 16], is seen as a key aid in realizing scalable model checks: instead of checking $M \models \phi$ for a large model M , construct an abstract model A from a compact specification of M such that $A \models \phi$ always implies $M \models \phi$ for certain kinds of properties ϕ . Predicate abstraction [15] partitions the state

* This work is in part financially supported by the DFG project *Refism* (FE 942/1-1)

space of a model M based on finitely many predicates of a suitable logic and then abstractly interprets [6] its transitions and labelings over that partition to render an abstract model A . If decidability of that logic is computable or can be approximated, this abstract model structure (if expressible in the logic) can be synthesized automatically with calls to a theorem prover. Predicate abstraction is highly successful as it allows the automatic computation of abstractions and an incremental refinement of the abstraction with new predicates in case that the current abstraction contains spurious information, i.e. that $A \models \phi$ erroneously suggests $M \models \phi$. Finding good heuristics for the choice of initial and refining predicates and proving relative completeness (i.e. possible termination) of this refinement process are two principal concerns in this line of research.

To enable finite-state model checking via abstractions in principle, we require that $M \models \phi$ implies $A \models \phi$ for some finite-state abstraction A of M ; as is customary, we call such A *feasible* abstractions. The existence of such a reduction for *all* properties of a given formal language is termed “completeness” in [10], a terminology we will adopt in this paper. Given completeness one could somehow find a magic abstraction even though the original problem $M \models \phi$ may be undecidable. For Kripke structures and formulas of linear-time temporal logic completeness has been shown by Kesten & Pnueli in [17], where models were augmented with progress monitors to allow abstractions to preserve liveness properties.

Branching-time temporal logics are needed in important application settings. We mention multiple system observers that prevent a “linearization of time,” and invariants of dynamical systems that mix different path quantifiers (e.g. whether all reachable states in a model of a biological system can reach a state from which a cyclic behavior is possible). Branching time may also aid in expressing process/environment interaction as seen in alternating-time temporal logic [1]. For branching time, Dams & Namjoshi have shown in [10] that Kripke structures are incomplete for Existential Computation Tree Logic (ECTL) but that completeness can be secured for the entire modal μ -calculus (including CTL and CTL*) if models are augmented to render tree-automata-like structures: focussed transition systems in [10], μ -automata in [11], etc. Completeness proofs share that they, in essence, construct a finite-state abstraction A of M with $A \models \phi$ from a proof that $M \models \phi$ holds. This completeness is an expressibility result and says nothing about how a feasible finite-state abstraction may be found.

Namjoshi [22] shows completeness for the modal μ -calculus through the abstraction of an alternating transition system $M \times \phi$, a product between a labeled transition system M and an alternating tree automata ϕ that represents the property to be checked. These abstractions use choice predicates at OR-states, and rank functions for monitoring progress. For this abstraction framework, and a set of predicates that contains the initially chosen ones and is closed under weakest preconditions, he shows that the abstraction is relatively complete iff (choices at OR-states are uniform and progress ranks are bounded). In [22] we therefore find a precise characterization of the kind of branching-time properties for which predicate abstraction can find a feasible abstraction within the abstraction framework of loc. cit.

Dams [7] considers a quality measure of an abstraction, precision, an optimality principle that is concerned with maximizing the number of properties being preserved

by the abstraction, given that its state space and model signature are fixed. In [22] precision is not covered.

In this paper we draw from the ideas in [7, 17, 22, 10] discussed above to develop an abstraction framework for Kripke structures and the modal μ -calculus (given here in the equivalent form of alternating tree automata) that meets the following objectives:

1. our framework is complete for the modal μ -calculus in the sense of [10]
2. our framework allows the construction of precise abstractions in the sense of [7]
3. our framework supports a notion of *ranked predicate abstraction*, a predicate abstraction that can deal with liveness properties as well
4. all abstractions, including those that prove completeness, are specified through ranked predicate abstractions that partition the state space they abstract
5. ranked predicate abstractions are incremental and thus open up the possibility of counter-example-guided abstraction refinement as familiar for linear time, and
6. ranked predicate abstractions are *confluent*: feasible abstractions, if they exist, can always be found in principle, regardless of the particular history of incremental refinements of an initially chosen abstraction

Although our models and methods are somewhat related to the work in [22], we highlight important differences. Abstractions in [22] are computed from a product of the model with the property to be checked. In contrast, our ranked predicate abstraction extends the state space partitions familiar from predicate abstraction with finitely many ranking functions and a set of slice predicates (Section 4). In [22] they find sufficient and necessary conditions for a feasible abstraction to be computed through predicate abstraction. In our framework we show that ranked predicate abstraction can indeed always express feasible abstractions (Section 6). We also show that our canonical ranked predicate abstraction is precise (Section 4) and that ranked predicate abstraction is incremental and confluent (Section 5).

The use of ranking functions, and their heuristics [9], for encoding fairness conditions is certainly not new. This use is seen in the aforementioned [17], in the compositional verification of liveness properties [12], and in the context of efficient complementation of automata [19]. We merely combine ranking functions with existing abstraction formalisms to prove desirable results for branching-time model checking.

2 Hypermixed Kripke structures

We define the models of interest, extensions of disjunctive modal transition systems [21]. Models with similar transition structure as those presented here, but with a different kind of acceptance condition, had already been proposed, and shown to be complete, in the technical report [13]. Without loss of generality, we won't consider action labels on models in this paper. Throughout, $|S|$ denotes the cardinality of a set S and $\mathbb{P}(S)$ denotes its power set.

Definition 1 (Models). *For a set of atomic propositions AP, a hypermixed Kripke structure M is a tuple $(S, R^-, R^+, L^-, L^+, (E_\ell, F_\ell)_{\ell \in \mathcal{L}})$ with finite \mathcal{L} such that*

- $(s \in)S$ is a set of states,

- $R^-, R^+ \subseteq S \times \mathbb{P}(S)$ the set of must- and may-transitions (respectively),
- $L^-, L^+: S \rightarrow \mathbb{P}(\text{AP})$ the must- and may-labelings (respectively) of states, and
- $(E_\ell, F_\ell)_{\ell \in \mathcal{L}}$ is a Streett acceptance condition with each (E_ℓ, F_ℓ) in $\mathbb{P}(S) \times \mathbb{P}(S)$.

We often refer to hypermixed Kripke structures as ‘models’. Furthermore, such a model is finite if $|S| + |\bigcup_{s \in S} L^-(s)| + |\text{AP} \setminus (\bigcup_{s \in S} L^+(s))|$ is finite.

Kripke structures have straightforward representations as hypermixed Kripke structures: let $R^- = R^+, L^- = L^+, \mathcal{L} = \{\}$, and ensure that $(s, D) \in R^-$ implies that D is a singleton. A hypermixed Kripke structure is depicted in Figure 1 and a Kripke structure is presented in Figure 2. The interpretation of the labelings L^- and L^+ is standard [7, 8]: $L^-(s)$ lists those atomic propositions that must hold in any refining states of s whereas $L^+(s)$ lists those propositions that may hold in some refinement of s . A must-transition $(s, D) \in R^-$ specifies that all refining states \check{s} of s in a Kripke structure \check{M} must have a transition $(\check{s}, \{\check{s}'\})$ in \check{M} such that \check{s}' refines *some* state in D [21]. Dually, a may-transition $(s, C) \in R^+$ specifies that all refining states \check{s} of s in a Kripke structure \check{M} may (but must not) have transitions in \check{M} of form $(\check{s}, \{\check{s}'\})$ such that \check{s}' refines *all* states in C . We formalize these intuitions in refinement games below.

The Streett acceptance condition for model M is a predicate \mathcal{A}_M that characterizes the allowed infinite sequences of states, those $(s_n)_{n \in \mathbb{N}}$ satisfying “for all $\ell \in \mathcal{L}$, set $\{n \in \mathbb{N} \mid s_n \in E_\ell\}$ is infinite or set $\{n \in \mathbb{N} \mid s_n \in F_\ell\}$ is finite”. For example in the model from Figure 1, the infinitely repeating sequence of states $(s_{00}^1 s_{10}^2)^\omega$ is accepted whereas $(s_{10}^0 s_{11}^1)^\omega$ is not, where the superscript 2 (resp., 1) denotes membership in E_ℓ (resp., F_ℓ). We chose a Streett condition over, say, a Rabin condition since its conjunctivity allows us to enforce *all* constraints of a ranking function; and since it guarantees that checking guarded formulas of the modal μ -calculus for such models is in NP as Player I will have a memoryless winning strategy.

We turn to defining abstraction between models through a refinement notion, using various acceptance conditions of regular games. Below we write π_i for the projection into the i -th component of an ordered tuple. Given a relation $\rho \subseteq B \times C$ with subsets $X \subseteq B$ and $Y \subseteq C$ we write $X.\rho$ for $\{c \in C \mid \exists b \in X: (b, c) \in \rho\}$ and $\rho.Y$ for $\{b \in B \mid \exists c \in Y: (b, c) \in \rho\}$ and abuse this notation whenever ρ is a function (viewed as a graph). For a sequence of tuples Φ we write $\Phi[i]$ for the sequence obtained from Φ through projection into the i -th coordinate. Let $\text{map}(f, \Phi)$ be the sequence obtained from Φ by applying function f to all elements of Φ in situ.

Definition 2 (Refinement).

1. Finite refinement plays for models M_1 and M_2 have the rules and winning conditions as stated in Table 1. An infinite play Φ is a win for Player I (the verifier) iff $[\mathcal{A}_{M_1}(\Phi[1]) \Rightarrow \mathcal{A}_{M_2}(\Phi[2])]$ holds; otherwise it is won by Player II (the refuter).
2. State $s_1 \in S_1$ refines $s_2 \in S_2$ (and then s_2 abstracts s_1) iff Player I has a winning strategy for all refinement plays started at (s_1, s_2) .
3. Model M_1 refines (is abstracted by) M_2 iff Player I has a strategy for the corresponding refinement game between M_1 and M_2 such that any state in S_1 is abstracted by some state in S_2 , and any state in S_2 is refined by some state in S_1 .

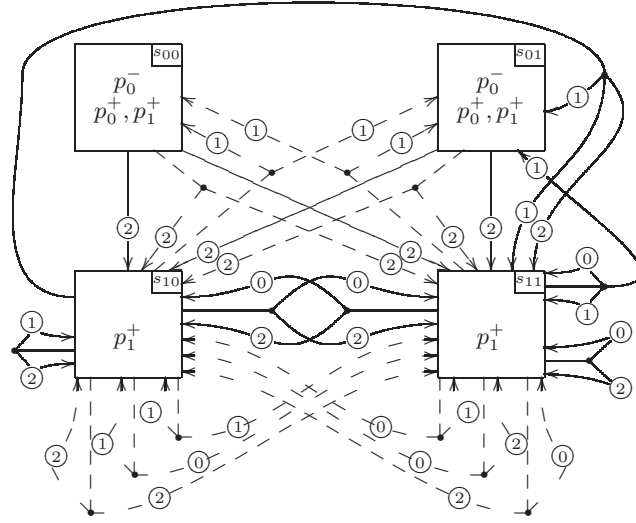


Fig. 1. A hypermixed Kripke structure. At state s , label p^- (resp., p^+) denotes $s \in L^-(p)$ (resp., $s \in L^+(p)$). Branching solid (resp., dashed) arrows model must-transitions (s, D) (resp., may-transitions (s, C)). Depicted states come in three versions — labeled with 0, 1, or 2 superscripts — that share outgoing transitions. Labels on transitions indicate the version of their source state. For example, the upper most depicted solid transition indicates $\{(s_{10}^i, \{s_{01}^1, s_{11}^2\}) \mid i \in \{0, 1, 2\}\} \subseteq R^-$. The state set of all versions labeled with 2, paired with the state set of all versions labeled with 1, yields the sole Streett acceptance condition of this model, here, $E = \{s_{00}^2, s_{01}^2, s_{10}^2, s_{11}^2\}$ and $F = \{s_{00}^1, s_{01}^1, s_{10}^1, s_{11}^1\}$

Our ranked predicate abstraction defined below will ensure item 3. above by definition. Since this paper presents techniques that render abstractions by construction, we are not concerned with the complexity of checking refinement per se. We note that our refinement between two Kripke structures coincides with bisimulation [23].

Example 1 (Abstraction). The model in Figure 1 is an abstraction of the model from Figure 2 where all three versions of s_{00} and s_{01} abstract \hat{s} , and all three versions of s_{10} and s_{11} abstract all states of the Kripke structure other than \hat{s} .

3 Sound satisfaction relation

We will present the modal μ -calculus in its equivalent form of alternating tree automata [25]. All results in this section have standard proofs.

Definition 3 (Tree automata). An alternating tree automaton $A = (Q_A, \delta_A, \Theta_A)$ has

- a finite, nonempty set of states $(q \in) Q_A$
- a transition relation δ_A mapping automaton states to one of the following forms, where q, q_1, q_2 are automaton states and $p \in \text{AP}$: $p \mid \neg p \mid q \mid q_1 \wedge q_2 \mid q_1 \vee q_2 \mid \text{EX } q \mid \text{AX } q$ and

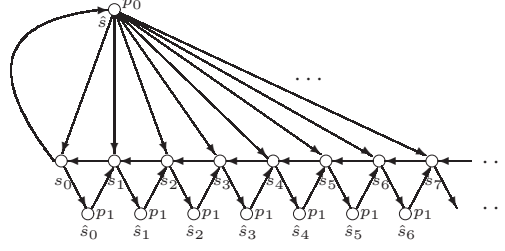


Fig. 2. A Kripke structure with $L^-(\hat{s}) = \{p_0\}$, $L^-(s_i) = \{\}$, and $L^-(\hat{s}_i) = \{p_1\}$ for all $i \geq 0$. Arrows $s \rightarrow s'$ denote $(s, \{s'\}) \in R^-$

L^- labeling: Player II chooses p from $L^-(s_2)$; Player I wins iff p is in $L^-(s_1)$

L^+ labeling: Player II chooses p from $AP \setminus L^+(s_2)$; Player I wins iff p is not in $L^+(s_1)$

R^- transition: Player II chooses a set of states $D'_2 \in \{s_2\}.R_2^-$; Player I responds with $D'_1 \in \{s_1\}.R_1^-$; Player II chooses $s'_1 \in D'_1$; Player I responds with $s'_2 \in D'_2$; the next configuration is (s'_1, s'_2)

R^+ transition: Player II chooses a set of states $C'_1 \in \{s_1\}.R_1^+$; Player I responds with $C'_2 \in \{s_2\}.R_2^+$; Player II chooses $s'_2 \in C'_2$; Player I responds with $s'_1 \in C'_1$; the next configuration is (s'_1, s'_2)

Table 1. Moves of refinement game at configuration (s_1, s_2) . Refinement plays are sequences of configurations generated thus

- an acceptance condition $\Theta_A: Q_A \rightarrow \mathbb{N}$ with finite image, where an infinite sequence of automata states is accepted iff the maximal acceptance number occurring infinitely often is even.

An alternating tree automaton is depicted in Figure 3. Throughout this paper, we assume without loss of generality [18] that all automata correspond to guarded formulas of the modal μ -calculus, i.e. that every cycle in the underlying graph of automaton A has to contain an element that is labeled with **EX** or **AX**. Also, for any bounded sequence \mathbf{n} of elements in \mathbb{N} we write $\text{sup}(\mathbf{n})$ for the largest m that occurs in \mathbf{n} infinitely often.

Definition 4 (Satisfaction).

- Finite satisfaction plays for model M and alternating tree automaton A have the rules and winning conditions as stated in Table 2. An infinite play Φ is a win for Player I iff $[\mathcal{A}_M(\Phi[1]) \Rightarrow \text{sup}(\text{map}(\Theta, \Phi[2]))]$ is even; otherwise it is won by Player II.
- Model M satisfies automaton A in configuration $(s, q) \in S \times Q$, written $(M, s) \models (A, q)$, iff Player I has a strategy for the corresponding satisfaction game between M and A such that Player I wins all satisfaction plays started at (s, q) with her strategy.

The acceptance condition for satisfaction plays between a model M and an automata A is a variant of those familiar from the literature: An infinite play Φ is a

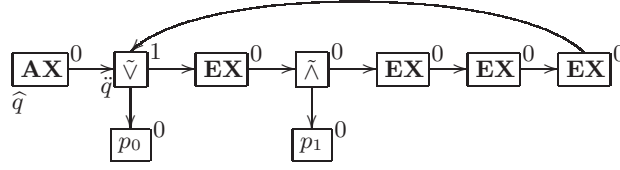


Fig. 3. An alternating tree automata. Accepting values are depicted next to states. At state \tilde{q} it expresses that there is a run that reaches a p_0 -state after $4n$ moves for some $n \geq 0$ (since the cycle has to be left to obtain an accepting sequence) such that p_1 always holds after $4m+1$ moves for every $m < n$ with $m \geq 0$. At \hat{q} , it expresses that after any transition the property expressed at \tilde{q} holds

p : Player I wins iff $p \in L^-(s)$

$\neg p$: Player I wins iff $p \notin L^+(s)$

q' : the next configuration is (s, q')

$q_1 \hat{\wedge} q_2$: Player II picks a q' from $\{q_1, q_2\}$; the next configuration is (s, q')

$q_1 \tilde{\vee} q_2$: Player I picks a q' from $\{q_1, q_2\}$; the next configuration is (s, q')

EX q' : Player I picks $D' \in \{s\}.R^-$; Player II picks $s' \in D'$; the next configuration is (s', q')

AX q' : Player II picks $C' \in \{s\}.R^+$; Player I picks $s' \in C'$; the next configuration is (s', q')

Table 2. Moves of satisfaction game at configuration (s, q) , specified through a case analysis on the value of $\delta(q)$. Satisfaction plays are sequences of configurations generated thus

win for Player I iff either the projection of Φ into the automata A is accepting in A ($\text{sup}(\text{map}(\Theta, \Phi[2]))$ is even) or the projection of Φ into M is non-accepting in M ($\neg \mathcal{A}_M(\Phi[1])$). We write $s \models q$ and Q , etc, whenever M and A are clear from the context. Note that \models applied to Kripke structures corresponds to the usual satisfaction relation.

Example 2 (Satisfaction game). For the model of Figure 4 and the automaton from Figure 3 we have $s_{00}^0 \models \hat{q}$: at the \hat{q} -state Player I chooses s_{10}^2 or s_{20}^2 in the **AX**-move, depending on which may-transition from s_{00}^0 is picked by Player II. In order to show \tilde{q} at s_{10}^2 or s_{20}^2 , Player I chooses the **EX**-automaton state in the $\tilde{\vee}$ -move, then she chooses the must-transition pointing to $\{s_{31}^0\}$, (if Player II picks the **EX**-state) Player I chooses the must-transition pointing to $\{s_{21}^0\}$, at the next **EX**-move she chooses the one pointing to $\{s_{10}^1, s_{20}^0\}$, at the next **EX**-move she chooses the one pointing to s_{01}^1 , respectively the one pointing to $\{s_{10}^1, s_{20}^1\}$. Then in the latter case, a cycle has been reached and the game continues as described before. So either p_0 is reached or a sequence that contains no state labeled with 2 but infinitely many states labeled with 1 is generated, which contradicts the Streett acceptance condition.

The winning conditions for the satisfaction game are Rabin conditions as they have form [Streett \Rightarrow RabinChain] which reduces to Rabin; so deciding $(M, s) \models (A, q)$ is in NP for finite-state models. We prove soundness of $(M, s) \models (A, q)$ as an approximation of the EXPTIME-hard relation which asks whether all pointed Kripke structures (\tilde{M}, \tilde{s}) that refine (M, s) satisfy A in (\tilde{s}, q) , the proof is completely standard. As usual,

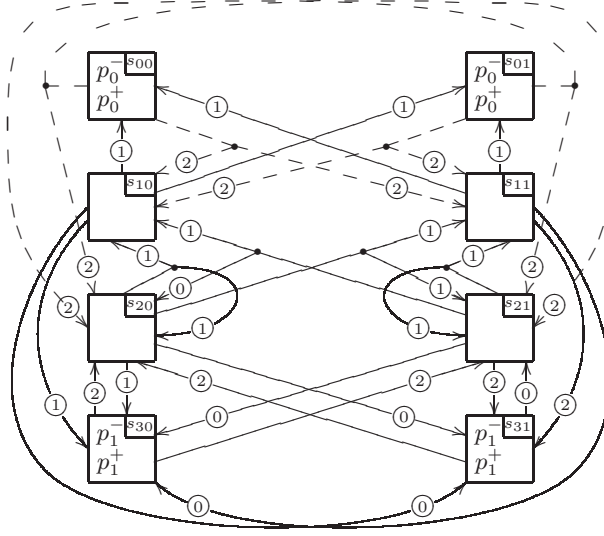


Fig. 4. Another hypermixed Kripke structure. For notational conventions we refer to Figure 1

$(M, s) \not\models (A, q)$ does not imply $(M, s) \models (\neg A, q)$ where $\neg A$ recognizes the complement of A .

Theorem 1 (Soundness). *Suppose s_1 refines s_2 . Then for any “guarded” automaton A and $q \in Q$, we have that $s_2 \models (A, q)$ implies $s_1 \models (A, q)$.*

4 Ranked predicate abstraction

Predicate abstraction computes a partition of a concrete state space by identifying states that have the same truth values for finitely many given formulas of some logic [15]. If that logic is decidable or if its decidability can be over-approximated, then model structure (e.g. a state transition relation) over this partition can be synthesized by means of such decision procedures without explicitly constructing the concrete model [15] — provided that model structure is expressible in the given logic.

We now adapt predicate abstraction to hypermixed Kripke structures and branching time and argue the suitability of that adaption. In doing so, we work with a function h that maps concrete states S to abstract states I . This function is derived from finitely many predicates $\phi_1, \phi_2, \dots, \phi_n$ by the equivalence relation $\equiv \subseteq S \times S$, given by $s \equiv s'$ iff (for all $1 \leq i \leq n$, $s \models \phi_i \Leftrightarrow s' \models \phi_i$). Then I is the set of equivalence classes of \equiv and $h(s)$ is defined to be the equivalence class of s .

Definition 5 (Ranked predicate abstraction). *A ranked predicate abstraction \aleph of a state space S is a tuple $(I, h, J, (\leq^k)_{k \in K}, \wp)$ where*

- $h: S \rightarrow I$ is a surjective function mapping concrete (S) to abstract (I) states

- J is a non-empty set of rank locations;
- for all $k \in K$, with K a (possibly empty) index set, $\leq^k \subseteq (S \times J) \times (S \times J)$ is a pre-order with well-founded irreflexive version $<^k$; and
- $\wp \subseteq \text{AP}$ is the set of slice predicates such that
- $|I| + |J| + |K| + |\wp|$ is finite.

Ranked predicate abstraction \aleph generalizes the familiar predicate abstraction [15] to the entire modal μ -calculus, and so to liveness properties in particular. In our approach

- precision and completeness require an acceptance condition: abstract runs that are infinitely descending and related (not necessarily state-wise) to concrete runs are rejected; “descending” is defined in terms of given pre-orders
- set J is used to allow more complex ranking pre-orders \leq^k
- set \wp is used to restrict the predicates of the model to those which occur in properties one wishes to check.

Example 3 (Ranked predicate abstraction). Two ranked predicate abstractions for state space S of the Kripke structure in Figure 2 are $\hat{\aleph} = (\{i_0, i_1\}, \hat{h}, \{\hat{j}, \hat{j}\}, (\leq^k)_{k \in \{0\}}, \{p_0\})$ and $\check{\aleph} = (\{i_0, i_1, i_2, i_3\}, \check{h}, \{\check{j}, \check{j}\}, (\leq^k)_{k \in \{0\}}, \{p_0, p_1\})$ such that $\hat{h}(\hat{s}) = \check{h}(\hat{s}) = i_0$, $\hat{h}(s_0) = \check{h}(s_0) = i_1$, and for $n \in \mathbb{N}$, $\hat{h}(s_{n+1}) = \hat{h}(\hat{s}_n) = i_1$, $\check{h}(s_{n+1}) = i_2$, $\check{h}(\hat{s}_n) = i_3$, and $(s', j') \leq^0 (s, j) \Leftrightarrow \omega(s', j') \leq \omega(s, j)$ for $s, s' \in S, j, j' \in \{\hat{j}, \check{j}\}$ where $\omega(\hat{s}, \hat{j}) = \omega(\hat{s}, \check{j}) = 0$, $\omega(s_n, \hat{j}) = \omega(\hat{s}_n, \check{j}) = n + 1$, and $\omega(s_n, \check{j}) = \omega(\hat{s}_n, \hat{j}) = n + 2$.

We point out that a nontrivial J is required to obtain completeness of the abstraction framework, see Proposition 1 in Section 6 below. The \aleph -abstraction game involves two hypermixed Kripke structures M_1 and M_2 where \aleph is a ranked predicate abstraction for the state space of M_1 . The objective of Player I is to show that M_1 is abstracted by M_2 up to \aleph , meaning that Player II can switch between states of M_1 that map to the same elements via h as long as no contradiction to the acceptance condition of M_1 or to the ranking functions of \aleph is produced. Therefore, the states of M_1 are represented in configurations via elements of $I \times J \times ((\mathcal{L} \cup K) \rightarrow \{0, 1, 2\})$, where J is under control of Player I such that soundness of the game is ensured, and $((\mathcal{L} \cup K) \rightarrow \{0, 1, 2\})$ is used to encode fairness constraints as follows: A Streett condition pair (E, F) , represented by an element $k \in K$, is encoded via a function into $\{0, 1, 2\}$ where value 2 indicates that a state of E , respectively, a move not preserving \leq^k happens; 1 indicates that a state of $F \setminus E$, respectively, a move that preserves $<^k$ happens; and 0 indicates that a state outside F or E , respectively, a move that preserves \leq^k (but not strictly so) happens. Formally $\Omega^{M, \aleph} : (S \times J)^2 \rightarrow ((\mathcal{L} \cup K) \rightarrow \{0, 1, 2\})$ is given by

$$\Omega_{(s', j', s, j)}^{M, \aleph}(x) = \begin{cases} 2 & \text{if } (x \in \mathcal{L} \ \& \ s' \in E_x) \text{ or } (x \in K \ \& \ (s', j') \not\leq^x (s, j)) \\ 1 & \text{if } (x \in \mathcal{L} \ \& \ s' \in F_x \setminus E_x) \text{ or } (x \in K \ \& \ (s', j') <^x (s, j)) \\ 0 & \text{otherwise} \end{cases}$$

where we assume throughout that K and \mathcal{L} are disjoint. For the completeness proof for $(M, s) \models (A, q)$ it is instructive to think of J as the set of automaton states Q_A . We write $\Phi(n)$ for the n -th configuration of play Φ .

L^- labeling: Player II chooses p from $L^-(s_2)$ and $s_1 \in h.\{i\}$; Player I wins iff $p \in L^-(s_1) \cap \wp$
 L^+ labeling: Player II chooses p from $\text{AP} \setminus L^+(s_2)$ and $s_1 \in h.\{i\}$; Player I wins iff $p \in \wp \setminus L^+(s_1)$

R^- transition: Player II chooses a set of states $D'_2 \in \{s_2\}.R_2^-$; Player I responds with $j' \in J$;
Player II chooses $s_1 \in h.\{i\}$; Player I responds with $D'_1 \in \{s_1\}.R_1^-$; Player II chooses $s'_1 \in D'_1$;
Player I responds with $s'_2 \in D'_2$; the next configuration is $(h(s'_1), j', \Omega_{(s'_1, j', s_1, j)}^{M, \aleph}, s'_2)$

R^+ transition: Player II chooses $s_1 \in h.\{i\}$ and $C'_1 \in \{s_1\}.R_1^+$; Player I responds with $C'_2 \in \{s_2\}.R_2^+$;
Player II chooses $s'_2 \in C'_2$; Player I responds with $s'_1 \in C'_1$ and $j' \in J$; the next configuration is $(h(s'_1), j', \Omega_{(s'_1, j', s_1, j)}^{M, \aleph}, s'_2)$

Table 3. Moves of \aleph -abstraction game at configuration $(i, j, g, s_2) \in I \times J \times ((\mathcal{L} \cup K) \rightarrow \{0, 1, 2\}) \times S_2$ where \aleph -abstraction plays are sequences of configurations generated thus

Definition 6 (Ranked predicate abstraction game). Let \aleph be a ranked predicate abstraction for the state space of M_1 .

- Finite \aleph -abstraction plays for models M_1 and M_2 have the rules and winning conditions as stated in Table 3. An infinite play Φ is a win for Player I iff [(for all $x \in \mathcal{L} \cup K$, $\sup((\pi_3(\Phi(n))(x))_{n \in \mathbb{N}})$ is even) $\Rightarrow \mathcal{A}_{M_2}(\Phi[4])$] holds; otherwise it is won by Player II.
- The model M_1 is \aleph -abstracted by M_2 iff Player I has a strategy for the corresponding \aleph -abstraction game between M_1 and M_2 such that for any $(i, j, g) \in I \times J \times ((\mathcal{L} \cup K) \rightarrow \{0, 1, 2\})$ there is $s_2 \in S_2$ (and, conversely, for all $s_2 \in S_2$ there is $(i, j, g) \in I \times J \times ((\mathcal{L} \cup K) \rightarrow \{0, 1, 2\})$) such that Player I wins with her strategy in all \aleph -abstraction plays started at (i, j, g, s_2) .

We show that \aleph -abstractions are indeed abstractions.

Theorem 2 (Consistency of ranked predicate abstractions). Let \aleph be a ranked predicate abstraction of the state space of M_1 . If M_1 is \aleph -abstracted by M_2 , then M_1 is abstracted by M_2 .

We now define an abstraction that is precise with respect to the \aleph -abstraction game.

Definition 7 (Precise ranked predicate abstraction). Let \aleph be a ranked predicate abstraction of the state space of M . The precise \aleph -abstraction M_{\aleph} of M is defined to be the model $(S_{\aleph}, R_{\aleph}^-, R_{\aleph}^+, L_{\aleph}^-, L_{\aleph}^+, (E_x^{\aleph}, F_x^{\aleph})_{x \in \mathcal{L} \cup K})$ where

$$\begin{aligned}
S_{\aleph} &= \{(i', j', \Omega_{(s', j', s, j)}^{M, \aleph}) \in I \times J \times ((\mathcal{L} \cup K) \rightarrow \{0, 1, 2\}) \mid i' = h(s')\} \\
R_{\aleph}^- &= \{((i, j, g), \tilde{D}) \mid \exists j' \in J, f: S \rightarrow \mathbb{P}(S) : (\forall s \in h.\{i\} : f(s) \in \{s\}.R^-) \& \\
&\quad \tilde{D} = \{(i', j', \Omega_{(s', j', s, j)}^{M, \aleph}) \mid s \in h.\{i\} \& s' \in h.\{i'\} \cap f(s)\}\} \\
R_{\aleph}^+ &= \{((i, j, g), \tilde{C}) \mid \exists s \in h.\{i\}, C \in \{s\}.R^+ : \\
&\quad \tilde{C} = \{(i', j', \Omega_{(s', j', s, j)}^{M, \aleph}) \mid j' \in J \& s' \in C \cap h.\{i'\}\}\} \\
L_{\aleph}^-(i, j, g) &= \wp \cap \bigcap_{s \in h.\{i\}} L^-(s) & L_{\aleph}^+(i, j, g) &= (\text{AP} \setminus \wp) \cup \bigcup_{s \in h.\{i\}} L^+(s) \\
E_x^{\aleph} &= \{(i, j, g) \mid g(x) = 2\} & F_x^{\aleph} &= \{(i, j, g) \mid g(x) = 1\}
\end{aligned}$$

In the \aleph -abstraction of M , the set of must-labelings (resp., may-labelings) is standard, except that it (resp., its complement) is restricted to predicates from \wp . The Streett conditions specify that value 1 must not occur infinitely often if value 2 occurs only finitely often. The state space is the reachable one occurring in the \aleph -abstraction game. The first component $h(s')$ corresponds to the abstract state, the second component j' corresponds to the rank location (the current automaton state if J is taken to be Q), and the third component corresponds to the acceptance encoded in the ranked predicate abstraction game.

A must-transition $((i, j, g), \tilde{D}) \in R_{\aleph}^-$ is determined by choosing must-transitions $(s, f(s)) \in R^-$ for every related concrete state $s \in h.\{i\}$ and taking a ranked predicate location $j' \in J$. Note that Player I has control of the ranked predicate location in the satisfaction game, so she can always respond with the reached automaton state if J is taken to be Q . Set \tilde{D} is the union of the abstract states in $f(s)$ for $s \in h.\{i\}$ combined with j' . Furthermore, the g -component of a target in \tilde{D} may vary depending on the considered witnesses s and s' in M . A may-transition $((i, j, g), \tilde{C}) \in R_{\aleph}^+$ is determined by taking a may-transition $(s, C) \in R^+$ for some s that is related to the abstract state i . The target set consists of all abstract states related to a state in C combined with any value from J . The g -component of a target in \tilde{C} may similarly vary depending on the witnesses for the abstract state in C .

Example 4. For $\hat{\aleph}$ and $\ddot{\aleph}$ of Example 3, the $\hat{\aleph}$ -abstraction of the model in Figure 2 is shown in Figure 1 and the $\ddot{\aleph}$ -abstraction of that same model is depicted in Figure 4. There the index of s_{ij} corresponds to the I -component (respectively J -component) and the g -component is encoded by the transition labels. These figures omit must-transitions that have matching must-transitions with a superset as target (these omission leads to refinement equivalent models). To enhance readability the must-transitions outgoing from states s_{00} and s_{01} as well as the outgoing may-transitions from other states are omitted in Figure 4.

We justify the adjective “precise” of Definition 7.

Theorem 3 (Precision). *The finite-state model M_{\aleph} of Definition 7 is a precise \aleph -abstraction of M , i.e.,*

- M_{\aleph} is a \aleph -abstraction of M and
- if M_2 is a \aleph -abstraction of M , then M_2 abstracts M_{\aleph} .

5 Incremental analysis

In the case that an abstraction obtained by ranked predicate abstraction does not satisfy a property of interest, techniques for abstraction-refinement that reuse already verified sub-properties are called for. Such a technique, a generalization of adding predicates in the predicate or Cartesian abstraction approach [2, 14], is introduced now:

Definition 8 (Extensions of ranked predicate abstractions). *Let \aleph_1 and \aleph_2 be ranked predicate abstractions of S . Then \aleph_1 is an extension of \aleph_2 if $h_2 = h_1 \circ h$ for some surjective function h , $J_2 \subseteq J_1$, $\wp_2 \subseteq \wp_1$, $K_2 \subseteq K_1$, and $\forall k \in K_2: \leq_1^k = \leq_2^k$.*

For example, the ranked predicate abstraction \aleph is an extension of $\widehat{\aleph}$, where \aleph and $\widehat{\aleph}$ are given in Example 3. Extensions always enable incremental analysis.

Theorem 4 (Incremental analysis). *Let the ranked predicate abstraction \aleph_1 be an extension of the ranked predicate abstraction \aleph_2 . Then M_{\aleph_1} is abstracted by M_{\aleph_2} .*

Extensions should be confluent in the following sense: if a ranked predicate abstraction \aleph_1 for the state space of some model M yields an abstraction M_{\aleph_1} satisfying the automaton (A, q) , then any ranked predicate abstraction \aleph_2 for the state space of M should be extendable to a ranked predicate abstraction \aleph such that M_{\aleph} also satisfies (A, q) . We define common extensions and show this desired confluence.

Definition 9 (Common extension). *Let \aleph_1 and \aleph_2 be ranked predicate abstractions for state space S , where we assume without loss of generality that K_1 and K_2 are disjoint. The ranked predicate abstraction $\aleph_1 \sqcap \aleph_2$ for S is $(S, h, h, J_1 \cup J_2, (\leq^k)_{k \in K_1 \cup K_2}, \wp_1 \cup \wp_2)$ where $h = \{(s, (i_1, i_2)) \in S \times (I_1 \times I_2) \mid s \in h_1 \cdot \{i_1\} \cap h_2 \cdot \{i_2\}\}$, and \leq^k is \leq_1^k if $k \in K_1$, but equals \leq_2^k if $k \in K_2$.*

Theorem 5 (Confluence of extensions). *Let \aleph_1 and \aleph_2 be ranked predicate abstractions for state space S . Then $\aleph_1 \sqcap \aleph_2$ is an extension of \aleph_1 and of \aleph_2 .*

6 Completeness

First we point out an issue of expressiveness: more than one rank location is needed in order to get a complete predicate abstraction.

Proposition 1 (Limited expressiveness). *There is no ranked predicate abstraction \aleph of the Kripke structure from Figure 2 such that its J is a singleton and $(M_{\aleph}, h(\hat{s})) \models (A, \hat{q})$ holds, where A is the automaton from Figure 3.*

We now construct ranked predicate abstractions that prove the desired completeness. Let $(M, s) \models (A, q)$ hold. The set of OR-states O_A and the set T_A of states that are targets of EX- or AX-states are defined:

$$\begin{aligned} O_A &= \{q \in Q \mid \exists q_1, q_2 \in S: \delta(q) = q_1 \tilde{\vee} q_2\} \\ T_A &= \{q' \in Q \mid \exists q \in Q: \delta(q) \in \{\mathbf{EX} q', \mathbf{AX} q'\}\}. \end{aligned}$$

Without loss of generality, the automaton state q that describes the property we are interested in is in T_A (otherwise add a fresh automaton state q' with $\delta(q') = \mathbf{EX} q$; thus q' is unreachable and won't interfere with satisfaction games at other automaton states).

A *choice function* for A is a function $c_A: O_A \rightarrow \{1, 2\}$. Let Ch_A be the set of all choice functions for A . Let θ be a memoryless strategy for Player I for the satisfaction game between M and A . Then $c_A^{\theta, s}$ is the choice function whose choices on any $q \in O_A$ agree with those of θ on (s, q) .

The ranked predicate abstraction that proves completeness with respect to a memoryless strategy θ is constructed as follows: States of M are equivalent iff

- they satisfy the same automaton states with respect to θ and

- θ behaves the same on every OR-state.

Relevant predicates are those that occur in the automaton. The set J is taken to be the set of automaton states T_A . The pre-orders \leq^k are derived from ranking functions corresponding to odd automaton acceptance numbers: roughly speaking, the ranking function $\omega^{\theta,k}$ is determined (if possible, otherwise a default value κ is chosen) by the least number of unfoldings necessary to guarantee that no further $2k + 1$ value can be reached by remaining below $2k + 2$. This is formalized by counting the unfoldings of function $\mathcal{U}_{\theta,k}$ applied to the empty set until the state of the model combined with the corresponding automaton state is obtained in the generated set (note that κ will be chosen such that it is always greater than any possible counting). Formally, $\mathcal{U}_{\theta,k}: (S \times Q) \rightarrow (S \times Q)$ is given by

$$\begin{aligned} \mathcal{U}_{\theta,k}(W) &= W \cup \{(s, q) \mid \forall (s_n, q_n)_{n \in \mathbb{N}} \in \xi_{(s,q)}^\theta, r \in \mathbb{N}: \\ &\quad \Theta(q_{r+1}) = 1 + 2k \Rightarrow ((s_{r+1}, q_{r+1}) \in W \text{ or } \exists r' \leq r: \Theta(q_{r'}) > 1 + 2k)\} \end{aligned}$$

with $\xi_{(s,q)}^\theta$ as set of all plays started in configuration (s, q) and played via strategy θ .

Definition 10 (Complete ranked predicate abstraction). *Let θ be a memoryless strategy for Player I for the satisfaction game between M and automaton A . Then the θ -ranked predicate abstraction is $\aleph_\theta = (S, h_\theta, h_\theta, T_A, (\leq_\theta^k)_{k \in K_\theta}, \wp_\theta)$, where*

$$\begin{aligned} K_\theta &= \{0, 1, \dots, \lfloor (\max\{\Theta(q) \mid q \in Q\} - 1)/2 \rfloor\} \text{ and } \kappa = |\mathbb{P}(\mathbb{P}(S \times Q))| \\ h_\theta &: S \rightarrow \mathbb{P}(T_A) \times \text{Ch}_A \text{ with } h_\theta(s) = (\{q \in T_A \mid \theta \text{ wins in } (s, q)\}, c_A^{\theta,s}) \\ \omega_{(s,q)}^{\theta,k} &= \min(\{\alpha \mid (s, q) \in \mathcal{U}_{\theta,k}^\alpha(\{\})\} \cup \{\kappa\}) \\ (s', q') \leq_\theta^k (s, q) &\Leftrightarrow \omega_{(s',q')}^{\theta,k} \leq \omega_{(s,q)}^{\theta,k} \\ \wp_\theta &= \{p \mid \exists q \in Q: \delta(q) \in \{p, \neg p\}\} \end{aligned}$$

Theorem 6 (Completeness). *Let M be a Kripke structure and θ be a memoryless strategy for Player I for $(M, s) \models (A, q)$ and \aleph_θ the θ -ranked predicate abstraction of M . Then $(M_{\aleph_\theta}, (h_\theta(s), q, g)) \models (A, q)$ holds whenever θ is winning for the satisfaction game at configuration (s, q) .*

7 Discussion

Fairness constraints in models, the Streett acceptance conditions in our paper, are required for securing completeness as the property language is powerful enough to express such constraints. Such completeness can already be proved if R^+ has type $S \times S$ instead of our $S \times \mathbb{P}(S)$. But then abstractions for a given \aleph may be less precise, and the completeness proof is likely to be harder; in fact, we don't know whether completeness for feasible abstractions restricted to state space partitions is then always realizable.

Our abstract models are closely related to the modal automata in [11], except that in our model

- only must-transitions point to OR-states, e.g., our must-transition (s, D) is graphically represented through an OR-state o that has exactly the outgoing transitions to all elements of D and s points to o ; and

- may-transitions point to AND-states via a similar graphical representation.

Note that AND-states do not exist in modal automata; in our approach AND-states allow more compact abstractions and simplify our completeness proof. We did consider using modal automata but found that a definition and proof of precision would be more complex than for our choice of model, as indicated in the previous paragraph.

We also considered using the focussed transition systems in [10] as an alternative complete abstraction framework. We decided against its use as one of our key objectives was to maximize the reuse of tried and tested methods in predicate abstraction, notably the partition of a concrete state space through predicates and the computation of abstract transitions based on the existence of transitions between states of such partitions. The focus and defocus operations in focussed transition systems seem to make it difficult to reason about the existence of transitions in this manner. This is related to the fact that the model checking game $F \models \phi$ for property ϕ and focussed transition systems F in [10] does not satisfy conjunction elimination and disjunction introduction; e.g. there are F , ϕ_1 , and ϕ_2 with $F \models \phi_1 \wedge \phi_2$ but $F \not\models \phi_1$. This also suggests that “most precise” abstractions may not exist or may be difficult to define for focussed transition systems.

Our completeness result, as all others, does not shed light on how to find feasible abstractions but it secures the existence of ranked predicate abstractions that are feasible, confluent, and incremental. So the design of an abstraction-refinement loop for predicate abstraction of branching time may be attainable in future work.

8 Conclusion

In this paper we developed an abstraction framework for Kripke structures that extends predicate abstraction to ranked predicate abstraction so that one can deal with all liveness properties as well. Specifically, whenever a Kripke structure M satisfies a property ϕ of the modal μ -calculus, there is a finite-state model computed through a ranked predicate abstraction that witnesses this truth, and so our framework is complete in the sense of Dams & Namjoshi [10]. We also proved that the abstractions synthesized in this way are precise in the sense of Dams [7]. Our ranked predicate abstractions correspond to state space partitions of the concrete model by definition. We demonstrated that these abstractions are incremental and confluent: new predicates may be added for abstraction-refinement, and feasible abstractions can be found no matter how, and how often, initial abstractions have been refined so far. In summary our results form a good foundation for the automated synthesis of abstractions and counter-example-guided abstraction-refinement for branching time, both subjects for future work.

References

1. R. Alur, Th. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM* 49(5):672–713, 2002.
2. Th. Ball, A. Podelski, and S. K. Rajamani. Boolean and Cartesian Abstraction for Model Checking C Programs. In *Proc. of TACAS'01*, LNCS 2031, pp. 268–283, Springer-Verlag, 2001.

3. G. Bruns and P. Godefroid. Model Checking Partial State Spaces with 3-Valued Temporal Logics. In *Proc. of CAV'99*, LNCS 1633, pp. 274–287, Springer-Verlag, 1999.
4. E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In *Logic of Programs Workshop*, LNCS 131, pp. 244–263. Springer-Verlag, 1981.
5. E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM TOPLAS* 16(5):1512–1542, 1994.
6. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs. In *Proc. of POPL'77*, pp. 238–252, ACM Press, 1977.
7. D. Dams. *Abstract interpretation and partition refinement for model checking*. PhD thesis, Technische Universiteit Eindhoven, The Netherlands, 1996.
8. D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM TOPLAS*, 19(2):253–291, 1997.
9. D. Dams, R. Gerth, and O. Grumberg. A Heuristic for the Automatic Generation of Ranking Functions. In *Proc. of the Workshop on Advances in Verification*, Chicago, July 2000.
10. D. Dams and K. Namjoshi. The Existence of Finite Abstractions for Branching Time Model Checking. In *Proc. of LICS'04*, pp. 335–344, IEEE Computer Society Press, 2004.
11. D. Dams and K. S. Namjoshi. Automata as Abstractions. In *Proc. of VMCAI'05*, LNCS 3385, pp. 216–232, Springer-Verlag, 2005.
12. Y. Fang, N. Piterman, A. Pnueli, and L. D. Zuck. Liveness with Incomprehensible Ranking. In *Proc. of TACAS'04*, LNCS 2988, pp. 482–496, Springer-Verlag, 2004.
13. H. Fecher and M. Huth. Complete abstractions through extensions of disjunctive modal transition systems. *Technical Report No. 0604, Institut für Informatik und Praktische Mathematik der Christian-Albrechts-Universität zu Kiel*, 31 pages, March 2006.
14. P. Godefroid, M. Huth, and R. Jagadeesan. Abstraction-based Model Checking using Modal Transition Systems. In *Proc. of CONCUR'01*, LNCS 2154, pp. 426–440, Springer-Verlag, 2001.
15. S. Graf and H. Saidi. Construction of abstract state graphs with PVS. In *Proc. of CAV'97*, LNCS 1254, pp. 72–83, Springer-Verlag, 1997.
16. M. Huth, R. Jagadeesan, and D. A. Schmidt. Modal transition systems: a foundation for three-valued program analysis. In *Proc. of ESOP'01*, LNCS 2028, pp. 155–169. Springer-Verlag, 2001.
17. Y. Kesten and A. Pnueli. Verification by Augmented Finitary Abstraction. *Inf. Comput.* 163(1):203–243, 2000.
18. D. Kozen. Results on the propositional μ -calculus. *Theoretical Computer Science* 27:333–354, 1983.
19. O. Kupferman and M. Y. Vardi. Complementation Constructions for Nondeterministic Automata on Infinite Words. In *Proc. of TACAS'05*, LNCS 3440, pp. 206–221, Springer-Verlag, 2005.
20. K. G. Larsen and B. Thomsen. A Modal Process Logic. In *Proc. of LICS'88*, pp. 203–210, IEEE Computer Society Press, 1988.
21. K. G. Larsen and L. Xinxin. Equation Solving Using Modal Transition Systems. In *Proc. of LICS'90*, pp. 108–117, IEEE Computer Society Press, 1990.
22. K. Namjoshi. Abstraction for Branching Time Properties. In *Proc. of CAV'03*, LNCS 2725, pp. 288–300, Springer-Verlag, 2003.
23. D. M. R. Park. Concurrency and automata on infinite sequences. In *Proc. of the 5th GI Conference*, LNCS 104, pp. 167–183, Springer-Verlag, 1989.
24. J. P. Quielle and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proc. of the 5th International Symposium on Programming*, 1981.
25. Th. Wilke. Alternating tree automata, parity games, and modal μ -calculus. *Bull. Soc. Math. Belg.*, 8(2):359–391, May 2001.