Dominance Testing via Model Checking

Ganesh Ram Santhanam, Samik Basu and Vasant Honavar
Department of Computer Science, Iowa State University, Ames, IA 50011, USA.
{gsanthan,sbasu,honavar}@cs.iastate.edu

Abstract

Dominance testing, the problem of determining whether an outcome is preferred over another, is of fundamental importance in many applications. Hence, there is a need for algorithms and tools for dominance testing. CP-nets and TCP-nets are some of the widely studied languages for representing and reasoning with preferences. We reduce dominance testing in TCP-nets to reachability analysis in a graph of outcomes. We provide an encoding of TCP-nets in the form of a Kripke structure for CTL. We show how to compute dominance using NuSMV, a model checker for CTL. We present results of experiments that demonstrate the feasibility of our approach to dominance testing.

Introduction

Many applications call for techniques for representing and reasoning about qualitative preferences over a set of alternatives that are described in terms of a set of variables. Because of the wide range of applications that involve reasoning with preferences, there is a need for developing tools for dominance testing that are useful in practice. CP-nets (Boutilier et al. 2004), TCP-nets (Brafman, Domshlak, and Shimony 2006) and Wilson's extensions to the above (Wilson 2004a; 2004b) are among the widely studied languages for representing preferences over not only the valuations of the variables, but also the relative importance among them. They use the *ceteris paribus* ("all else being equal") interpretation of preference to reason with such preferences.

Ceteris paribus semantics induces a graph, the *induced preference graph* (Boutilier et al. 2004; Brafman, Domshlak, and Shimony 2006); and an outcome α is said to dominate another outcome β if there exists a path consisting of successively worsening outcomes in this graph from α to β. With the exception of special cases such as CP-nets with tree or polytree structured conditional dependencies (Boutilier et al. 2004; Brafman, Domshlak, and Shimony 2006), dominance testing has been shown to be PSPACE-complete (Goldsmith et al. 2008). However, experience with other hard problems such as boolean satisfiability (SAT) suggests that it is often possible to realize acceptable performance in practice, using implementations that take advantage of specialized data structures and algorithms.

Hence, this paper explores a novel approach to dominance testing that leverages the state-of-the-art techniques in *model checking* (Clarke, Emerson, and Sistla 1986; Queille and Sifakis 1982). Our approach reduces dominance testing to reachability analysis in a graph of outcomes. Specifically, we formalize the ceteris paribus semantics of preferences in terms of a direct and succinct representation of preference semantics using *Kripke structures* (Clarke, Grumberg, and Peled 2000) that encode preferences over outcomes as reachability within a graph of outcomes. In this setting, we reduce dominance testing to the satisfiability of corresponding temporal formulas in the model. We provide a translation from TCP-nets to the Kripke model specification language of a widely used model checker NuSMV (Cimatti et al. 2002). We demonstrate how a *proof* of dominance can be automatically generated whenever the dominance holds. This approach allows us to take advantage of the state-of-the-art model checkers that provide optimized computation of dominance using specialized data structures and algorithms. We present results of experiments that demonstrate the feasibility of this approach to dominance testing: Dominance queries over preference specifications involving 20 or more variables are answered within a few seconds. While the discussion in this paper is restricted to TCP-nets, our approach to dominance testing via model checking can be used for any preference formalism whose semantics is given in terms of properties over a graph of outcomes.

Preference Language

Let $V = \{X_i\}$ be a set of variables, each with a domain D_i. An outcome $\alpha \in \mathcal{O}$ is a complete assignment to all the variables, denoted by the tuple $\alpha := \langle \alpha(X_1), \alpha(X_2), \ldots, \alpha(X_m) \rangle$ such that $\alpha(X_i) \in D_i$ for each $X_i \in V$. The set of all possible outcomes is given by $\mathcal{O} = \prod_{X_i \in V} D_i$. We consider a preference language for specifying: (a) conditional intra-variable preferences $\succ_i$ that are strict partial orders (i.e., irreflexive and transitive relations) over D_i; and (b) conditional relative importance preferences $\rhd$ that are strict partial orders over V.

CP-nets (Boutilier et al. 2004) use a compact graphical model to specify conditional intra-variable preferences $\succ_i$ over a set of variables V. Each node in the graph corre-

sponds to a variable $X_i \in V$, and each *dependency* edge (X_i, X_j) in the graph captures the fact that the intra-variable preference $\succ_j$ with respect to variable X_j is dependent (or conditioned) on the valuation of X_i. For any variable X_j, the set of variables $\{X_i : (X_i, X_j)$is an edge$\}$ that influence $\succ_j$ are called the *parent* variables, denoted $Pa(X_j)$. Each node X_i in the graph is associated with a *conditional preference table* (CPT) that maps all possible assignments to the parents $Pa(X_i)$ to a total order over D_i. An *acyclic* CP-net is one that does not contain any dependency cycles.

TCP-nets (Brafman, Domshlak, and Shimony 2006) extend CP-nets by allowing additional edges (X_i, X_j) to be specified, describing the relative importance among variables ($X_i \rhd X_j$). Each relative importance edge could be either unconditional (directed edge) or conditioned on a set of *selector* variables (analogous to parent variables in the case of intra-variable preferences). Each edge (X_i, X_j) describing conditional relative importance is undirected and is associated with a table (analogous to the CPT) mapping each assignment of the selector variables to either $X_i \rhd X_j$ or vice versa. Figure 1 illustrates a TCP-net.

Ceteris Paribus Semantics

A formal semantics in terms of the *ceteris paribus* interpretation for preference languages involving conditional intra-variable and relative importance preferences was given by Brafman et al. in (Brafman, Domshlak, and Shimony 2006).

Definition 1 (Worsening flipping sequence: adapted from (Brafman, Domshlak, and Shimony 2006))**.** *A sequence of outcomes $\alpha = \gamma_1, \gamma_2, \cdots \gamma_{n-1}, \gamma_n = \beta$ such that*

$$\alpha = \gamma_1 \succ^\circ \gamma_2 \succ^\circ \cdots \succ^\circ \gamma_{n-1} \succ^\circ \gamma_n = \beta$$

*is a **worsening flipping sequence** with respect to a set of preference statements if and only if, for $1 \leq i < n$[1], either*

1. (*V-flip*) *outcome γ_i is different from the outcome γ_{i+1} in the value of exactly one variable X_j, and $\gamma_i(X_j) \succ_j \gamma_{i+1}(X_j)$, or*
2. (*I-flip*) *outcome γ_i is different from the outcome γ_{i+1} in the value of exactly **two** variables X_j and X_k, $\gamma_i(X_j) \succ_j \gamma_{i+1}(X_j)$, and $X_j \rhd X_k$.*

The V-flips are induced directly by the conditional intra-variable preferences $\succ_i$, and the I-flips are additional flips induced by the relative importance $\rhd$ over variables in conjunction with $\succ_i$. Note that the notion of an I-flip in this definition revises the one presented in (Brafman, Domshlak, and Shimony 2006) in order to accurately reflect the semantics of $\succ^\circ$ [2]. Furthermore, this definition adapts the original definition such that flips are worsening rather than improving. Given a TCP-net N and a pair of outcomes α and β, Brafman et al. have shown that $\alpha \succ^\circ \beta$ with respect to N if and only if there is an worsening flipping sequence with respect to N from α to β.

An issue with dominance testing with respect to $\succ^\circ$ is that the interpretation of relative importance statements is *pairwise*, as illustrated by Wilson in (Wilson 2004b; 2004a). In this case, I-flips allow *only two* variables to be flipped at a time. On the other hand, Wilson's extended semantics (Wilson 2004b; 2004a) (denoted $\succ^\blacksquare$) defines a worsening I-flip to allow multiple variables to be changed at a time in order to produce a worse outcome. This generalizes the search for *flipping* sequences to a search for *swapping*[3] sequences.

Definition 2 (Worsening flipping sequence with revised I-flip: adapted from (Wilson 2004b; 2004a))**.** *A sequence of outcomes $\alpha = \gamma_1, \gamma_2, \cdots \gamma_{n-1}, \gamma_n = \beta$ such that*

$$\alpha = \gamma_1 \succ^\blacksquare \gamma_2 \succ^\blacksquare \cdots \succ^\blacksquare \gamma_{n-1} \succ^\blacksquare \gamma_n = \beta$$

*is a **worsening flipping sequence** with respect to a set of preference statements if and only if, for $1 \leq i < n$, either*

1. (*V-flip*) *as in Definition 1*
2. (*I-flip*) *outcome γ_i is different from the outcome γ_{i+1} in the value of variables X_j and $X_{k_1}, X_{k_2}, \cdots X_{k_n}$, $\gamma_i(X_j) \succ_j \gamma_{i+1}(X_j)$, and $X_j \rhd X_{k_1}, X_j \rhd X_{k_2}, \cdots, X_j \rhd X_{k_n}$.*

Given a TCP-net N and a pair of outcomes α and β, according to Wilson's semantics we say that N entails $\alpha \succ^\blacksquare \beta$ iff there is a worsening flipping sequence with respect to N from α to β.

Dominance Testing via Model Checking

We now proceed to describe our approach to dominance testing using model checking. The key observation behind our approach is that dominance of α over β is given in terms of the reachability of the worse outcome (β) from the preferred outcome (α) in the *induced preference graph* (Boutilier et al. 2004; Brafman, Domshlak, and Shimony 2006) that captures the preference semantics.

Definition 3. *Given a TCP-net N over a set of variables V, the* induced preference graph *$\delta(N) = G(A, E)$ is constructed as follows. The nodes A correspond to the set of all possible outcomes, i.e., complete assignments to all variables in V, and each directed edge $(\alpha, \beta) \in E$ corresponds to either a V-flip or an I-flip as dictated by the chosen semantics (Definitions 1, 2).*

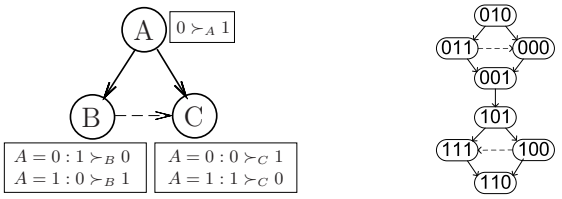
An outcome α dominates β with respect to N if and only if the node corresponding to β in $\delta(N)$ is reachable from α. Note that $\delta(N)$ is guaranteed to be acyclic because it represents an irreflexive and transitive dominance relation.

Example. Consider a TCP-net N of three binary variables, namely $\{A, B, C\}$ as shown in Figure 1(a). $\succ_B$ and $\succ_C$ depend on the valuations A (solid directed edges), and the nodes are annotated with the respective CPTs. $B \rhd C$ is denoted by a dotted edge from B to C. Figure 1(b) shows the

[1] We assume the dominance relation is irreflexive and transitive in this paper.

[2] Specifically, Definition 1 relaxes the stronger requirement (see Definition 13 in (Brafman, Domshlak, and Shimony 2006)) that "$\gamma_{i+1}(X_j) \succ_j \gamma_i(X_j)$ and $\gamma_i(X_k) \succ_k \gamma_{i+1}(X_k)$" to a weaker requirement that "$\gamma_{i+1}(X_j) \succ_j \gamma_i(X_j)$" – based on a personal communication exchanged by the authors with Ronen Brafman.

[3] To simplify the terminology, we will henceforth use the term *flipping* sequence to refer to Wilson's *swapping* sequence as well.



(a) (b)

Figure 1: (a) TCP-net N; (b) Transitive Reduction of $\delta(N)$

transitive reduction of the corresponding induced preference graph $\delta(N)$.

The above formulation of dominance testing in a TCP-net N in terms of verifying reachability properties in the corresponding induced preference graph $\delta(N)$ allows us to take advantage of the state-of-the-art approaches to model checking. This approach to dominance testing involves addressing two questions: (a) How to encode the induced preference graph $\delta(N)$ as an input graph to a model checker (we use NuSMV (Cimatti et al. 2002)), and (b) How to express a query regarding the dominance of an outcome (α) with respect to another (β) in the form of a test of reachability of β from α in the corresponding graph.

The preference variables of the TCP-net are mapped to the state variables of the model in a model checker. The V-flips and the I-flips (Definitions 1 and 2) are directly encoded as transitions in the *Kripke* structures for the language of the model checker. This ensures that the state space explored by the model checker corresponds to $\delta(N)$.

Dominance queries over the TCP-nets are then modeled as temporal logic properties (in CTL (Clarke, Grumberg, and Peled 2000)) over the state space of the model. This allows us to take advantage of all the specialized data structures (e.g., BDDs) and algorithms available in the model checking engine to efficiently verify the satisfiability of the corresponding temporal logic properties. In order to check whether an outcome α dominates the outcome β, we query the model checker with a CTL formula φ such that there is a model of φ if and only if α dominates β. If the dominance does hold, then the model checker can be used to obtain a proof of dominance, i.e., a worsening flipping sequence from α to β.

Kripke Structure Encoding of TCP-net Preferences

We now proceed to describe how TCP-net[4] preferences can be encoded in a *Kripke* structure (Clarke, Grumberg, and Peled 2000).

Definition 4 (Kripke Structure)**.** *A* Kripke *structure is a tuple $\langle S, S_0, T, L\rangle$ where S is a set of states described by the valuations of a set of propositional variables P, $S_0 \subseteq S$ is a set of* initial *states, $T \subseteq S \times S$ is a transition relation inducing directed edges between states such that $\forall s \in S : \exists s' \in S : (s, s') \in T$, and $L : S \rightarrow 2^P$ is a labeling function such that $\forall s \in S : L(s)$ is the set of propositions that are* true *in s.*

Given a TCP-net N over a set $V = \{X_1, \ldots X_n\}$ of variables, a Kripke structure K_N corresponding to the induced preference graph $\delta(N)$ can be constructed as follows.

1. The states S are defined by the valuations of propositions $P = V \cup \{h_i | X_i \in V\}$, where each h_i is a binary variable indicating whether or not the value of X_i can change in a transition,

$$h_i = \begin{cases} 0 & \text{if value of } X_i \textbf{ must not} \text{ change in a transition in the Kripke structure } K_N \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

   The change variables h_i defined above will be used, as will be shown later, to construct the state space S of the Kripke structure K_N that encodes the induced preference graph $\delta(N)$. Note that each outcome α in $\delta(N)$ corresponds to a set $S^\alpha = \{s | s_{\downarrow V} = \alpha\}$ of states, where $s_{\downarrow V}$ denotes the *projection* of a state s described by P onto the set of variables $V \subseteq P$. By this construction, the various states in S^α differ precisely in the valuations of the change variables, and since there are $|V|$ change variables, $|S^\alpha| = 2^{|V|}$. The start state(s) S_0 of K_N are specified based on the dominance query (as described later); and the labeling function L is defined such that for any state $s \in S$, $L(s)$ corresponds to the set of variables that are 'true' in s.

2. The transition relation T is defined as follows, with $s(X_i)$ and $s'(X_i)$ denoting the valuation of the corresponding variable X_i in states s and s' respectively. For any two states $s, s' \in S$, define $(s, s') \in T$ (denoted $s \rightarrow s'$) by the rules:

   i. (V-flip)

$$s \rightarrow s' \Leftarrow \begin{cases} \exists X_i \in V : s(h_i) = 1 \wedge s(X_i) \succ_i s'(X_i) \\ \wedge \forall X_j \in V \setminus \{X_i\} : s(h_j) = 0 \wedge \\ s(X_j) = s'(X_j) \end{cases}$$

   ii. (I-flip)

$$s \rightarrow s' \Leftarrow \begin{cases} \exists X_i \in V : s(h_i) = 1 \wedge s(X_i) \succ_i s'(X_i) \\ \wedge \exists W \subseteq V \setminus \{X_i\} : \forall X_j \in W : X_i \rhd X_j \\ \wedge \forall X_k \in V \setminus (W \cup \{X_i\}) : \\ s(h_k) = 0 \wedge s(X_k) = s'(X_k) \end{cases}$$

   iii. $s \rightarrow s' \Leftarrow \forall X_i \in V : s(X_i) = s'(X_i)$

In the above encoding of the Kripke structure, transition rules $2(i) - (iii)$ are exhaustive, thus satisfying the requirement that in a Kripke structure all states should have outgoing transitions. Transitions effected through the rules $2(i)$ and $2(ii)$ correspond to valid worsening V-flips and I-flips respectively according to Wilson's semantics[5] (Definition 2). Therefore, all possible edges E corresponding to V-flips and I-flips of the induced preference graph

[4]To simplify the presentation, we will restrict our discussion to TCP-nets over binary variables, although our approach can be extended to variables with other domains as well.

[5]Brafman et al. semantics (Definition 1) can be similarly encoded with minor changes to the definition of the transition relation $2(ii)$ in the Kripke structure.

$\delta(N) = G(A, E)$ are captured by the above transition relation. $2(iii)$ allows only transitions from a state s to those states s' that agree with s on all variables in V. We now establish the main theorem which forms the basis for our approach to dominance testing via model checking.

Remark. The definition of a V- or an I- flip from α to β (Definitions 1 and 2) requires the equality of α and β with respect to some of the variables in V. Since NuSMV does not allow the specification of a transition by constraining the destination state variables, we use h_i to control the allowed changes to each variable $X_i \in V$. Observe that the variables h_i are allowed to take any value in the destination state of any transition (see rules $2(i) - (iii)$). This allows the model checker to explore all possible V-/I-flips from any given outcome.

Theorem 1. *Given a TCP-net N, and the corresponding Kripke structure $K_N = \langle S, S_0, T, L\rangle$ (constructed from $\delta(N) = G(A, E)$ as described above),*

1. $\forall \alpha, \beta : (\alpha, \beta) \in E \Rightarrow \exists s \rightarrow s' : s_{\downarrow V} = \alpha \wedge s'_{\downarrow V} = \beta$
2. $\forall s, s' \in S : s \rightarrow s' \wedge s_{\downarrow V} \neq s'_{\downarrow V} \Rightarrow \exists (\alpha, \beta) \in E$

Proof. For the first part, let $(\alpha, \beta) \in E$. Since $\delta(N)$ is a cycle-free graph over distinct outcomes, $\alpha \neq \beta$. Further, $(\alpha, \beta) \in E$ requires the existence of either a V-flip or I-flip from α to β by Definitions 2 and 3. By construction of the Kripke structure, there exist sets S^α and S^β of states such that $\forall s^\alpha \in S^\alpha : s^\alpha_{\downarrow V} = \alpha$ and $\forall s^\beta \in S^\beta : s^\beta_{\downarrow V} = \beta$ respectively. Therefore, $\forall s^\alpha \in S^\alpha, s^\beta \in S^\beta : (s^\alpha_{\downarrow V}, s^\beta_{\downarrow V}) \in E$. By the definition of the valuations of the change variables in Equation (1) and the transition rules $2(i)$ and $2(ii)$, it follows that $\exists s^\alpha \in S^\alpha, s^\beta \in S^\beta : s^\alpha \rightarrow s^\beta$.

For the second part, let $s, s' \in S : s \rightarrow s' \wedge s_{\downarrow V} \neq s'_{\downarrow V}$. Since $s \neq s'$, $2(iii)$ is not applicable, and hence, it must be the case that the conditions in the right hand side of $2(i)$ or $2(ii)$ is satisfied. This in turn implies that the transition $s \rightarrow s'$ is due to a V-flip or an I-flip, i.e., $(s_{\downarrow V}, s'_{\downarrow V}) \in E$. □

Encoding K_N in NuSMV

For the TCP-net N specified in Figure 1, the encoding of the corresponding Kripke structure $\delta(N)$ that is provided as input to the NuSMV model checker (Cimatti et al. 2002) is shown in Figure 2. The `VAR` construct declares the binary preference variables (a, b, c) and the corresponding change variables (ha, hb, hc), and `ASSIGN` defines the transition rules for each variable in the form of the guards and corresponding next state valuations (as per rules $2(i) - (iii)$ in our construction). For example, the V-flips corresponding to variable `b` when `a` $= 0$ is encoded as follows: "**if** `b` $= 1 \&$ `a` $= 0 \&$ `ha` $= 0 \&$ `hb` $= 1 \&$ `hc` $= 0$ **then** `b` $= 0$ in the next state". The I-flips induced by `b` $\rhd$ `c` are specified by guarded transitions allowing (1) `b`'s valuation to change regardless of whether `c`'s valuation changes or not, and (2) `c`'s valuation to change in a transition whenever `b`'s valuation changes.

```
MODULE main
VAR a:{0,1}; b:{0,1}; c:{0,1};
     ha:{0,1};  hb:{0,1};  hc:{0,1};
ASSIGN --init(a):=1; init(b):=1; init(c):=1;
  init( ha):=0; init( hb):=0; init( hc):=0;
  next(a) :=
     case    -- conditional preferences:
        a=0 &  hc=0 &  hb=0 &  ha=1 : 1;
        1 : a;
     esac;
  next(b) :=
     case    -- conditional preferences:
        b=1 & a=0 &  hc=0 &  hb=1 &  ha=0 : 0;
        b=0 & a=1 &  hc=0 &  hb=1 &  ha=0 : 1;
             -- relative importance: b imp. than c
        b=1 & a=0 &  hb=1 &  ha=0 : 0;
        b=1 & a=0 &  hb=0 &  ha=0 : 1;
        b=0 & a=1 &  hb=1 &  ha=0 : 1;
        b=0 & a=1 &  hb=0 &  ha=0 : 0;
        1 : b;
     esac;
  next(c) :=
     case    -- conditional preferences:
        c=0 & a=0 &  hc=1 &  hb=0 &  ha=0 : 1;
        c=1 & a=1 &  hc=1 &  hb=0 &  ha=0 : 0;
             -- relative importance: c less imp. than b
        ((b=1 & a=0) | (b=0 & a=1))
                          &  hb=1 &  ha=0 &  hc=1 : !c;
        ((b=1 & a=0) | (b=0 & a=1))
                          &  hb=1 &  ha=0 &  hc=0 : c;
        1 : c;
     esac;
```

Figure 2: Listing of Kripke encoding in NuSMV

Computing Dominance

Given a Kripke structure K_N that encodes the induced preference graph of a TCP-net N, determining whether α dominates β in N can be reduced to verifying appropriate temporal properties in CTL (see (Clarke, Grumberg, and Peled 2000)). Specifically, the CTL formula $\varphi_\alpha \rightarrow \mathsf{EF}\varphi_\beta$ is used to check whether α dominates β. In the formula, φ_α and φ_β are conjunctions of the assignments to the variables in V in α and β respectively. A state in the Kripke structure is said to satisfy the above formula if and only if when the state satisfies φ_α (i.e., valuations of variables of V in that state correspond to those in α), there exists a path or a sequence of transitions $s^\alpha = s_1 \rightarrow s_2 \rightarrow \cdots \rightarrow s_n = s^\beta$ (s.t. $s^\alpha_{\downarrow V} = \alpha$ and $s^\beta_{\downarrow V} = \beta$) such that $n > 1$. In short, a state in the Kripke structure K_N corresponding to $\delta(N)$ satisfies the above CTL formula if and only if α dominates β with respect to N (Theorem 1). We will use the model checker NuSMV to verify the satisfiability of a CTL formula $\varphi_\alpha \rightarrow \mathsf{EF}\varphi_\beta$.

Example. For the TCP-net N in Figure 1, the dominance of $\alpha = \langle \mathsf{a} = 0, \mathsf{b} = 1, \mathsf{c} = 1\rangle$ over $\beta = \langle \mathsf{a} = 1, \mathsf{b} = 0, \mathsf{c} = 0\rangle$ corresponds to the satisfiability of the CTL formula $\varphi : (\mathsf{a} = 0 \& \mathsf{b} = 1 \& \mathsf{c} = 1 \rightarrow \mathsf{EF}(\mathsf{a} = 1 \& \mathsf{b} = 0 \& \mathsf{c} = 0))$. Note that NuSMV asserts that φ is verified only if *every* initial state satisfies φ. Therefore, we initialize X_i to $\alpha(X_i)$ to restrict the start states to S^α in the encoded Kripke structure. We also initialize all the change variables

h_i to 0, so that transitions corresponding to all possible V-flips and I-flips from α are explored by the model checker. In NuSMV, the satisfiability of φ can be verified by the specification $\mathsf{SPEC}\varphi$, and the verification returns 'true' in our example, thereby establishing that the outcome $\alpha = \langle \mathsf{a} = 0, \mathsf{b} = 1, \mathsf{c} = 1\rangle$ dominates $\beta = \langle \mathsf{a} = 1, \mathsf{b} = 0, \mathsf{c} = 0\rangle$.

Extracting a Proof of Dominance

We can use the NuSMV model checker to obtain a proof that an outcome α dominates another outcome β (i.e., a worsening flipping sequence from α to β) as follows. Suppose α dominates β. This implies that the CTL formula $\varphi : \varphi_\alpha \rightarrow \mathsf{EF}\varphi_\beta$ holds. Hence, in this case if we provide the formula $\neg\varphi$ (i.e., $\neg(\varphi_\alpha \rightarrow \mathsf{EF}\varphi_\beta)$) as input to the model checker, the model checker will return 'false', and provide us with the sequence of states (as below) corresponding to the worsening flipping sequence from α to β.

In our example, since $\alpha = \langle \mathsf{a} = 0, \mathsf{b} = 1, \mathsf{c} = 1\rangle$ dominates $\beta = \langle \mathsf{a} = 1, \mathsf{b} = 0, \mathsf{c} = 0\rangle$, when input the formula

$$\mathsf{SPEC}\ !\ (\mathsf{a} = 0\ \&\ \mathsf{b} = 1\ \&\ \mathsf{c} = 1 \\ \mathsf{->}\ \mathsf{EF}\ (\mathsf{a} = 1\ \&\ \mathsf{b} = 0\ \&\ \mathsf{c} = 0))$$

NuSMV returns the sequence: $(0,1,1) \rightarrow (0,0,0) \rightarrow (1,0,0)$ as shown below.

```
-> State: 1.1 <-        -> State: 1.2 <-        ha = 1
  a = 0                   hb = 1                hb = 0
  b = 1                   hc = 1                hc = 0
  c = 1
  ha = 0                -> State: 1.3 <-      -> State: 1.4 <-
  hb = 0                  b = 0                 a = 1
  hc = 0                  c = 0                 ha = 0
```

In the above, the transition from state 1.1 to 1.2 is effected by transition rule $2(iii)$; that from state 1.2 to 1.3 by rule $2(ii)$; and that from state 1.3 to 1.4 by rule $2(i)$.

Experiments and Results

We now describe the results of experiments that show the feasibility of our approach to dominance testing with respect to (a) the running time for dominance queries, and (b) scalability as a function of the number of preference variables that can be efficiently handled in practice. We generated random preference networks[6] by varying (a) number of variables s_v; (b) the maximum number of rows in a variable's conditional preference table s_c; and (c) number of relative importance edges s_r in the TCP-net. We restricted the maximum degree of each node (with respect to conditional dependencies) in the generated preference networks to 5. We conducted three sets of experiments:

1. *Set 1: Polytree CP-nets with Binary Variables.* We fixed $s_d = 2$ and $s_r = 0$, and varied s_v from 10 to 30 for each $s_c \in \{5, 10, 15\}$. In this case we generated only CP-nets that were singly connected, i.e., polytree structured CP-nets. We generated 20 preference networks for each choice of the parameters s_v and s_c. Using each resulting preference network, we evaluated 20 dominance queries by picking distinct pairs of outcomes at random.
2. *Set 2: Acyclic CP-nets with Binary Variables.* We fixed $s_d = 2$ and $s_r = 0$, and varied s_v from 4 to 27 for each $s_c \in \{5, 10, 15\}$. We generated only CP-nets with acyclic dependencies. We generated 20 preference networks for each choice of the parameters s_v and s_c. Using each resulting preference network, we evaluated 20 dominance queries by picking distinct pairs of outcomes at random.
3. *Set 3: TCP-nets with Binary Variables.* We fixed $s_d = 2$, varied s_v from 4 to 20 for each $s_c \in \{5, 10, 15\}$ and for each $s_r \in \{s_v/2, s_v, 2 \times s_v\}$. We ensured that the generated TCP-nets are satisfiable (see (Brafman, Domshlak, and Shimony 2006)), by requiring that the union of the conditional dependency and relative importance edges in the generated TCP-net induces an acyclic TCP-net. We generated 5 preference networks for each choice of the parameters s_v, s_c and s_r. Using each resulting preference network, we evaluated 5 dominance queries by picking distinct pairs of outcomes at random.

Set	s_v	s_c	s_r	Avg.Time(sec)
Set 1	10	$5-15$	-	0.03
Set 1	20	$5-15$	-	0.10
Set 1	30	$5-15$	-	11.24
Set 2	$4-20$	$5-15$	-	0.10
Set 2	$21-27$	$5-15$	-	12.52
Set 3	$4-15$	$5-15$	$s_v/2$	0.09
Set 3	$4-15$	$5-15$	s_v	0.25
Set 3	$4-15$	$5-15$	$2 \times s_v$	0.68
Set 3	$16-20$	$5-15$	$s_v/2$	1.51
Set 3	$16-20$	$5-15$	s_v	13.96
Set 3	$16-20$	$5-15$	$2 \times s_v$	< 60.0

Table 1: Running times of dominance queries for TCP-nets with binary variables

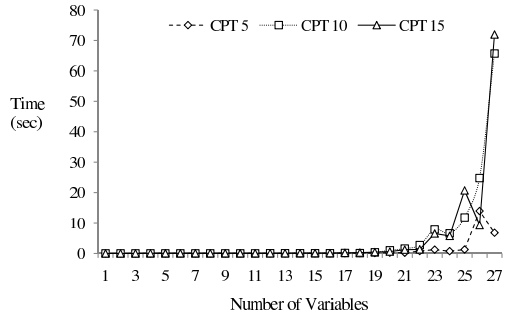


Figure 3: Running time vs. number of variables

The results (summarized in Table 1) show that our approach to dominance testing answers dominance queries within a few seconds for TCP-nets with upto 20 to 30 variables. In the case of polytree structured CP-nets (Set 1), the dominance queries were answered within a few seconds for upto 30 variables and dependent variables with CPTs with upto 15 rows. For arbitrary acyclic CP-nets (Set 2), our so-

[6]We modified the code developed by Cozman et al. (Ide et al. 2004) for generating random Bayesian networks to generate random TCP-nets.

lution performs well for CP-nets with close to 30 variables with CPTs with upto 15 rows. For TCP-nets (Set 3), our solution is efficient for upto 20 variables.

In summary, our experiments show that dominance testing using model checking is quite feasible for TCP-nets that are large enough to capture preferences in many practical applications.

Summary and Discussion

We have described, to the best of our knowledge, the first practical solution to the problem of determining whether an outcome dominates another with respect to a set of qualitative preferences. Our approach relies on a reduction of the dominance testing problem to reachability analysis in a graph of outcomes. We have provided an encoding of TCP-nets in the form of a Kripke structure for CTL.

We have shown how to: (a) directly and succinctly encode preference semantics as a Kripke structure; (b) compute dominance by verifying CTL temporal properties against this Kripke structure; and (c) generate a proof of dominance. We have shown how to compute dominance using NuSMV, a model checker for CTL. The results of our experiments demonstrate the feasibility of this approach to dominance testing. This approach to dominance testing via model checking allows us to take advantage of continuing advances in model-checking.

Although our treatment focused on acyclic CP-nets and TCP-nets, our approach can be applied to any preference language for which the semantics is given in terms of the satisfiability of graph properties (including GCP-nets, cyclic CP-nets and the language due to Wilson). Our approach can also be used for reasoning tasks other than dominance testing such as finding whether a given outcome is the *least* (*or most*) *preferred* among all the outcomes.

In our experiments we have reported running times for dominance testing that are intended merely to demonstrate the feasibility of our approach. The running times are averages taken over multiple dominance queries over sets of randomly generated preference networks. The running time could depend on many factors such as the structure of the preference network, the number of preference variables and the CPT size, in addition to the dominance query itself. Hence, it remains to be studied how the running times and memory usage of our solution approach are affected by such factors.

Although we have used the NuSMV model checker in our implementation, any model checker that accepts a Kripke structure as input can be used to realize our approach to dominance testing. Hence, it should be possible to take advantage of specialized techniques that have recently been developed to improve the performance of model checkers (Kahlon, Wang, and Gupta 2009; Cook and Sharygina 2005; Ciardo, Lüttgen, and Siminiceanu 2001; Wang 2000; Biere et al. 1999).

Acknowledgments

This work is supported in parts by NSF grants CNS0709217, CCF0702758 and IIS0711356.

References

Biere, A.; Cimatti, A.; Clarke, E. M.; and Zhu, Y. 1999. Symbolic model checking without bdds. In *Proc. Tools and Algorithms for the Construction and Analysis of Systems (TACAS), LNCS 1579*, 193–207. Springer.

Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res. (JAIR)* 21:135–191.

Brafman, R. I.; Domshlak, C.; and Shimony, S. E. 2006. On graphical modeling of preference and importance. *J. Artif. Intell. Res. (JAIR)* 25:389424.

Ciardo, G.; Lüttgen, G.; and Siminiceanu, R. 2001. Saturation: an efficient iteration strategy for symbolic state space generation. In *Proc. Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 328–342. Springer-Verlag.

Cimatti, A.; Clarke, E.; Giunchiglia, E.; Giunchiglia, F.; Pistore, M.; Roveri, M.; Sebastiani, R.; and Tacchella, A. 2002. NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In *Proc. Intl. Conf. on Computer-Aided Verification*. Copenhagen, Denmark: Springer.

Clarke, E. M.; Emerson, E. A.; and Sistla, A. P. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.* 8(2):244–263.

Clarke, E.; Grumberg, O.; and Peled, D. 2000. *Model Checking*. MIT Press.

Cook, B., and Sharygina, N. 2005. Symbolic model checking for asynchronous boolean programs. In *SPIN*, 75–90.

Goldsmith, J.; Lang, J.; Truszczynski, M.; and Wilson, N. 2008. The computational complexity of dominance and consistency in cp-nets. *J. Artif. Intell. Res. (JAIR)* 33:403–432.

Ide, J. S.; Cozman, F. G.; Ramos, F. T.; and Politcnica, E. 2004. Generation of random bayesian networks with constraints on induced width, with application to the average analysis of d-connectivity, quasi-random sampling, and loopy propagation. In *In Proc. of European Conf. on Art. Intel.*, 323–327.

Kahlon, V.; Wang, C.; and Gupta, A. 2009. Monotonic partial order reduction: An optimal symbolic partial order reduction technique. In *Proc. of Intl. Conf. on Computer Aided Verification*, 398–413. Springer-Verlag.

Queille, J.-P., and Sifakis, J. 1982. Specification and verification of concurrent systems in CESAR. In *International Symposium on Programming*, 337 – 351. Springer Verlag.

Wang, F. 2000. Efficient data structure for fully symbolic verification of real-time software systems. In *Proc. of Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), LNCS 1785*, 157–171. Springer-Verlag.

Wilson, N. 2004a. Consistency and constrained optimisation for conditional preferences. In *ECAI*, 888–894.

Wilson, N. 2004b. Extending cp-nets with stronger conditional preference statements. In *AAAI*, 735–741.