Automating Termination in Model Checking Modulo Theories

A. Carioni,[1] **S. Ghilardi**,[1] and S. Ranise[2]

[1]Università degli Studi di Milano

[2]FBK -Trento

Genova, September 29, 2011

General features

- Our puropose is a declarative re-interpretation of classical achievements from infinite state model checking literature.
- We use the classical approach to safety problems by Abdulla et al. (seminal paper [ACJT - LICS 96]) through backward reachability.
- Our declarative approach does not require any special formal model, just suitable logical encoding.
- SMT-solvers are the main algorithmic engine.
- Advantages: expressivity, flexibility, (relatively) easy implementation and maintenance.
- Advanced engineering of SMT-solvers guarantees efficiency.
- Large applications spectrum: distributed, timed, fault tolerant, but also sequential systems.

General features

- Our puropose is a declarative re-interpretation of classical achievements from infinite state model checking literature.
- We use the classical approach to safety problems by Abdulla et al. (seminal paper [ACJT - LICS 96]) through backward reachability.
- Our declarative approach does not require any special formal model, just suitable logical encoding.
- SMT-solvers are the main algorithmic engine.
- Advantages: expressivity, flexibility, (relatively) easy implementation and maintenance.
- Advanced engineering of SMT-solvers guarantees efficiency.
- Large applications spectrum: distributed, timed, fault tolerant, but also sequential systems.

General features

- Our puropose is a declarative re-interpretation of classical achievements from infinite state model checking literature.
- We use the classical approach to safety problems by Abdulla et al. (seminal paper [ACJT - LICS 96]) through backward reachability.
- Our declarative approach does not require any special formal model, just suitable logical encoding.
- SMT-solvers are the main algorithmic engine.
- Advantages: expressivity, flexibility, (relatively) easy implementation and maintenance.
- Advanced engineering of SMT-solvers guarantees efficiency.
- Large applications spectrum: distributed, timed, fault tolerant, but also sequential systems.

General features

- Our puropose is a declarative re-interpretation of classical achievements from infinite state model checking literature.
- We use the classical approach to safety problems by Abdulla et al. (seminal paper [ACJT - LICS 96]) through backward reachability.
- Our declarative approach does not require any special formal model, just suitable logical encoding.
- SMT-solvers are the main algorithmic engine.
- Advantages: expressivity, flexibility, (relatively) easy implementation and maintenance.
- Advanced engineering of SMT-solvers guarantees efficiency.
- Large applications spectrum: distributed, timed, fault tolerant, but also sequential systems.

General features

- Our puropose is a declarative re-interpretation of classical achievements from infinite state model checking literature.
- We use the classical approach to safety problems by Abdulla et al. (seminal paper [ACJT - LICS 96]) through backward reachability.
- Our declarative approach does not require any special formal model, just suitable logical encoding.
- SMT-solvers are the main algorithmic engine.
- Advantages: expressivity, flexibility, (relatively) easy implementation and maintenance.
- Advanced engineering of SMT-solvers guarantees efficiency.
- Large applications spectrum: distributed, timed, fault tolerant, but also sequential systems.

General features

- Our puropose is a declarative re-interpretation of classical achievements from infinite state model checking literature.
- We use the classical approach to safety problems by Abdulla et al. (seminal paper [ACJT - LICS 96]) through backward reachability.
- Our declarative approach does not require any special formal model, just suitable logical encoding.
- SMT-solvers are the main algorithmic engine.
- Advantages: expressivity, flexibility, (relatively) easy implementation and maintenance.
- Advanced engineering of SMT-solvers guarantees efficiency.
- Large applications spectrum: distributed, timed, fault tolerant, but also sequential systems.

General features

- Our puropose is a declarative re-interpretation of classical achievements from infinite state model checking literature.
- We use the classical approach to safety problems by Abdulla et al. (seminal paper [ACJT - LICS 96]) through backward reachability.
- Our declarative approach does not require any special formal model, just suitable logical encoding.
- SMT-solvers are the main algorithmic engine.
- Advantages: expressivity, flexibility, (relatively) easy implementation and maintenance.
- Advanced engineering of SMT-solvers guarantees efficiency.
- Large applications spectrum: distributed, timed, fault tolerant, but also sequential systems.

Well-Structured Transition Systems (WSTS)

Seminal paper [ACJT - LICS96]

$$(S, \tau, \preceq)$$

- S: set of states;
- $\tau = \{\rightarrow_\lambda \subseteq S \times S\}_\lambda$: labelled directed graph;
- $\preceq$: well quasi ordering
- each τ_λ is monotonic:

$$\begin{array}{ccc} s_1 & \preceq & s_2 \\ \downarrow_\lambda & & \downarrow_\lambda \\ s_3 & \preceq \exists & s_4 \end{array}$$

Well-Structured Transition Systems (WSTS)

Seminal paper [ACJT - LICS96]

$$(S, \tau, \preceq)$$

- S: set of states;
- $\tau = \{\rightarrow_\lambda \subseteq S \times S\}_\lambda$: labelled directed graph;
- $\preceq$: well quasi ordering
- each τ_λ is monotonic:

$$\begin{array}{ccc} s_1 & \preceq & s_2 \\ \downarrow_\lambda & & \downarrow_\lambda \\ s_3 & \preceq \exists & s_4 \end{array}$$

Well-Structured Transition Systems (WSTS)

Seminal paper [ACJT - LICS96]

$$(S, \tau, \preceq)$$

- S: set of states;
- $\tau = \{\rightarrow_\lambda \subseteq S \times S\}_\lambda$: labelled directed graph;
- $\preceq$: well quasi ordering
- each τ_λ is monotonic:

$$\begin{array}{ccc} s_1 & \preceq & s_2 \\ \downarrow_\lambda & & \downarrow_\lambda \\ s_3 & \preceq \exists & s_4 \end{array}$$

Well-Structured Transition Systems (WSTS)

Seminal paper [ACJT - LICS96]

$$(S, \tau, \preceq)$$

- S: set of states;
- $\tau = \{\rightarrow_\lambda \subseteq S \times S\}_\lambda$: labelled directed graph;
- $\preceq$: well quasi ordering
- each τ_λ is monotonic:

$$\begin{array}{ccc} s_1 & \preceq & s_2 \\ \downarrow_\lambda & & \downarrow_\lambda \\ s_3 & \preceq\ \exists & s_4 \end{array}$$

Well-Structured Transition Systems (WSTS)

Seminal paper [ACJT - LICS96]

$$(S, \tau, \preceq)$$

- S: set of states;
- $\tau = \{\rightarrow_\lambda \subseteq S \times S\}_\lambda$: labelled directed graph;
- $\preceq$: well quasi ordering
- each τ_λ is monotonic:

$$\begin{array}{ccc} s_1 & \preceq & s_2 \\ \downarrow_\lambda & & \downarrow_\lambda \\ s_3 & \preceq\ \exists & s_4 \end{array}$$

Well-Structured Transition Systems (WSTS)

Seminal paper [ACJT - LICS96]

$$(S, \tau, \preceq)$$

- S: set of states;
- $\tau = \{\rightarrow_\lambda \subseteq S \times S\}_\lambda$: labelled directed graph;
- $\preceq$: well quasi ordering
- each τ_λ is monotonic:

$$\begin{array}{ccc} s_1 & \preceq & s_2 \\ \downarrow_\lambda & & \downarrow_\lambda \\ s_3 & \preceq \exists & s_4 \end{array}$$

Well-Structured Transition Systems

- Set of unsafe states represented by an upset K:

$$s \in K \wedge s \preceq s' \rightarrow s' \in K$$

- Monotonicity implies that the pre-image of an upset is still an upset

$$Pre(\tau, K) \quad := \quad \{s \mid \exists \lambda \, \exists s' \, (s \xrightarrow{\lambda} s') \wedge s' \in K\}$$

- Since $\preceq$ is a wqo, upsets can be finitely represented by their *finitely many* minimal elements

Backward Reachability for WSTS

Checking that a set K of unsafe states is (un-)reachable from a set I of initial states

function BReach(K)
  $i \longleftarrow 0$; $BR^0(\tau, K) \longleftarrow K$; $K^0 \longleftarrow K$
  if $BR^0(\tau, K) \cap I \neq \emptyset$ **then return** unsafe
  repeat
    $K^{i+1} \longleftarrow \mathrm{Pre}(\tau, K^i)$
    $BR^{i+1}(\tau, K) \longleftarrow BR^i(\tau, K) \cup K^{i+1}$
    if $BR^{i+1}(\tau, K) \cap I \neq \emptyset$ **then return** unsafe
    else $i \longleftarrow i + 1$
  until $BR^{i+1}(\tau, K) \subseteq BR^i(\tau, K)$
  return safe
end

Backward Reachability for WSTS

Checking that a set K of unsafe states is (un-)reachable from a set I of initial states

function BReach(K)
  $i \longleftarrow 0$; $BR^0(\tau, K) \longleftarrow K$; $K^0 \longleftarrow K$
  if $BR^0(\tau, K) \cap I \neq \emptyset$ **then return** unsafe
  repeat
    $K^{i+1} \longleftarrow \mathrm{Pre}(\tau, K^i)$
    $BR^{i+1}(\tau, K) \longleftarrow BR^i(\tau, K) \cup K^{i+1}$
    if $BR^{i+1}(\tau, K) \cap I \neq \emptyset$ **then return** unsafe
    else $i \longleftarrow i + 1$
  until $BR^{i+1}(\tau, K) \subseteq BR^i(\tau, K)$
  return safe
end

Termination and Empirical Success

- Since $\preceq$ is a wqo, the algorithm terminates.
- Good success in the verification of safety properties of a variety of systems: broadcast protocols, cache coherence protocols, lossy channels systems, parameterized timed automata, etc.
- Extensions to cases in which $\preceq$ is not a wqo often terminate 'in practice'.

OUR GOAL: to get a declarative formulation of all this and to obtain an efficient backward reachability analysis by using state-of-the-art SMT solving for both safety and fix-point checking.

Termination and Empirical Success

- Since $\preceq$ is a wqo, the algorithm terminates.
- Good success in the verification of safety properties of a variety of systems: broadcast protocols, cache coherence protocols, lossy channels systems, parameterized timed automata, etc.
- Extensions to cases in which $\preceq$ is not a wqo often terminate 'in practice'.

OUR GOAL: to get a declarative formulation of all this and to obtain an efficient backward reachability analysis by using state-of-the-art SMT solving for both safety and fix-point checking.

Termination and Empirical Success

- Since $\preceq$ is a wqo, the algorithm terminates.
- Good success in the verification of safety properties of a variety of systems: broadcast protocols, cache coherence protocols, lossy channels systems, parameterized timed automata, etc.
- Extensions to cases in which $\preceq$ is not a wqo often terminate 'in practice'.

OUR GOAL: to get a declarative formulation of all this and to obtain an efficient backward reachability analysis by using state-of-the-art SMT solving for both safety and fix-point checking.

Termination and Empirical Success

- Since $\preceq$ is a wqo, the algorithm terminates.
- Good success in the verification of safety properties of a variety of systems: broadcast protocols, cache coherence protocols, lossy channels systems, parameterized timed automata, etc.
- Extensions to cases in which $\preceq$ is not a wqo often terminate 'in practice'.

OUR GOAL: to get a declarative formulation of all this and to obtain an efficient backward reachability analysis by using state-of-the-art SMT solving for both safety and fix-point checking.

Background on Array-based Systems

By a *theory* we mean here a pair $T = (\Sigma, \mathcal{C})$, where Σ is a first-order signature and $\mathcal{C}$ is a class of Σ-structures (called the models of T).

- Topology of the parameterised system: theory $T_I = (\Sigma_I, \mathcal{C}_I)$
  E.g.: $\mathcal{C}_I$ consists of all (finite) sets, linear orders, forests/trees, graphs, ...
- Data manipulated by the parameterised system: theories $T_{E_i} = (\Sigma_{E_i}, \mathcal{C}_{E_i})$ Usually $\mathcal{C}_{E_i}$ contains just one structure: integers, reals, Booleans, control locations, ... The T_{E_i} are joined to form a single multi-sorted theory called T_E.
- We assume the availability of SMT solvers deciding the satisfiability of quantifier-free formulae modulo T_I and the T_{E_i}.

Background on Array-based Systems

By a *theory* we mean here a pair $T = (\Sigma, \mathcal{C})$, where Σ is a first-order signature and $\mathcal{C}$ is a class of Σ-structures (called the models of T).

- Topology of the parameterised system: theory $T_I = (\Sigma_I, \mathcal{C}_I)$
  E.g.: $\mathcal{C}_I$ consists of all (finite) sets, linear orders, forests/trees, graphs, ...
- Data manipulated by the parameterised system: theories $T_{E_i} = (\Sigma_{E_i}, \mathcal{C}_{E_i})$ Usually $\mathcal{C}_{E_i}$ contains just one structure: integers, reals, Booleans, control locations, ... The T_{E_i} are joined to form a single multi-sorted theory called T_E.
- We assume the availability of SMT solvers deciding the satisfiability of quantifier-free formulae modulo T_I and the T_{E_i}.

Background on Array-based Systems

By a *theory* we mean here a pair $T = (\Sigma, \mathcal{C})$, where Σ is a first-order signature and $\mathcal{C}$ is a class of Σ-structures (called the models of T).

- Topology of the parameterised system: theory $T_I = (\Sigma_I, \mathcal{C}_I)$
  E.g.: $\mathcal{C}_I$ consists of all (finite) sets, linear orders, forests/trees, graphs, ...
- Data manipulated by the parameterised system: theories $T_{E_i} = (\Sigma_{E_i}, \mathcal{C}_{E_i})$ Usually $\mathcal{C}_{E_i}$ contains just one structure: integers, reals, Booleans, control locations, ... The T_{E_i} are joined to form a single multi-sorted theory called T_E.
- We assume the availability of SMT solvers deciding the satisfiability of quantifier-free formulae modulo T_I and the T_{E_i}.

Background on Array-based Systems

By a *theory* we mean here a pair $T = (\Sigma, \mathcal{C})$, where Σ is a first-order signature and $\mathcal{C}$ is a class of Σ-structures (called the models of T).

- Topology of the parameterised system: theory $T_I = (\Sigma_I, \mathcal{C}_I)$
  E.g.: $\mathcal{C}_I$ consists of all (finite) sets, linear orders, forests/trees, graphs, ...
- Data manipulated by the parameterised system: theories $T_{E_i} = (\Sigma_{E_i}, \mathcal{C}_{E_i})$ Usually $\mathcal{C}_{E_i}$ contains just one structure: integers, reals, Booleans, control locations, ... The T_{E_i} are joined to form a single multi-sorted theory called T_E.
- We assume the availability of SMT solvers deciding the satisfiability of quantifier-free formulae modulo T_I and the T_{E_i}.

Background on Array-Based Systems

- the sort INDEX is constrained by T_I;
- the sorts $\texttt{ELEM}_i$ are constrained by T_{E_i};
- the sorts $\texttt{ARRAY}_i$ represents arrays of $\texttt{ELEM}_i$ defined on INDEX;
- the 'read' operations $\_[\_]_i$ are added to $\Sigma_I \cup \Sigma_{E_i}$;
- the class of models of A_I^E consists of the multi-sorted structures whose reducts are models of T_I, T_{E_i} and the sorts ARRAY are interpreted as the set of total functions from indexes to elements and the read operations are interpreted as function application.

Background on Array-Based Systems

- An array-based system on A_I^E with array state variables $a = a_1, \ldots, a_s$ is a pair of formulae:

$$\mathcal{S} = \langle I(a), \tau(a, a') \rangle.$$

- A state of an array-based system is an assignment to the variables a in a model of A_I^E
- A safety problem for $\mathcal{S}$ is the following: given a formula $K(a)$, is

$$I(a_0) \wedge \tau(a_0.a_1) \wedge \cdots \wedge \tau(a_{n-1}, a_n) \wedge K(a_n)$$

A_I^E-satisfiable for some n?

Background on Array-Based Systems

- An array-based system on A_I^E with array state variables $a = a_1, \ldots, a_s$ is a pair of formulae:

$$\mathcal{S} = \langle I(a), \tau(a, a') \rangle.$$

- A state of an array-based system is an assignment to the variables a in a model of A_I^E
- A safety problem for $\mathcal{S}$ is the following: given a formula $K(a)$, is

$$I(a_0) \wedge \tau(a_0.a_1) \wedge \cdots \wedge \tau(a_{n-1}, a_n) \wedge K(a_n)$$

A_I^E-satisfiable for some n?

Background on Array-Based Systems

- An array-based system on A_I^E with array state variables $a = a_1, \ldots, a_s$ is a pair of formulae:

$$\mathcal{S} = \langle I(a), \tau(a, a') \rangle.$$

- A state of an array-based system is an assignment to the variables a in a model of A_I^E
- A safety problem for $\mathcal{S}$ is the following: given a formula $K(a)$, is

$$I(a_0) \wedge \tau(a_0.a_1) \wedge \cdots \wedge \tau(a_{n-1}, a_n) \wedge K(a_n)$$

A_I^E-satisfiable for some n?

Backward Reachability for Array-based Systems

Idea: recast symbolically the backward reachability algorithm

function BReach(K)
  $i \longleftarrow 0$; $BR^0(\tau, K) \longleftarrow K$; $K^0 \longleftarrow K$
  if A_I^E-check$(BR^0(\tau, K) \wedge I) = \text{sat}$ **then return** unsafe
  repeat
    $K^{i+1} \longleftarrow \text{Pre}(\tau, K^i)$
    $BR^{i+1}(\tau, K) \longleftarrow BR^i(\tau, K) \vee K^{i+1}$
    if A_I^E-check$(BR^{i+1}(\tau, K) \wedge I) = \text{sat}$ **then return** unsafe
    else $i \longleftarrow i + 1$
  until A_I^E-check$(\neg(BR^{i+1}(\tau, K) \rightarrow BR^i(\tau, K)) = \text{unsat}$
  return safe
end

But this is problematic... unless right formats for I, τ, K are found!

Format for initialization formulae

Proposed format for I: $\forall^I$-formulae

$$\forall \underline{i}\ \phi(\underline{i}, a[\underline{i}])$$

where $\underline{i}$ is a tuple of variables of sort `INDEX` and ϕ is a quantifier-free $\Sigma_I \cup \Sigma_E$-formula[1]

Example

For instance, the formula $\forall i.\ a[i] = \texttt{idle}$ says that all processes are in state `idle`.

$\forall^I$-formulae can also be used to express invariants

[1] If $\underline{i} = i_1, \ldots, i_n$, then $a[\underline{i}]$ is the tuple of terms $a_1[i_1], \ldots, a_s[i_n]$.

Format for initialization formulae

Proposed format for I: $\forall^I$-formulae

$$\forall \underline{i}\ \phi(\underline{i}, a[\underline{i}])$$

where $\underline{i}$ is a tuple of variables of sort `INDEX` and ϕ is a quantifier-free $\Sigma_I \cup \Sigma_E$-formula[1]

Example

For instance, the formula $\forall i.\ a[i] = \texttt{idle}$ says that all processes are in state `idle`.

$\forall^I$-formulae can also be used to express invariants

[1] If $\underline{i} = i_1, \ldots, i_n$, then $a[\underline{i}]$ is the tuple of terms $a_1[i_1], \ldots, a_s[i_n]$.

Format for initialization formulae

Proposed format for I: $\forall^I$-formulae

$$\forall \underline{i}\ \phi(\underline{i}, a[\underline{i}])$$

where $\underline{i}$ is a tuple of variables of sort `INDEX` and ϕ is a quantifier-free $\Sigma_I \cup \Sigma_E$-formula[1]

Example

For instance, the formula $\forall i.\ a[i] = \texttt{idle}$ says that all processes are in state `idle`.

$\forall^I$-formulae can also be used to express invariants

[1] If $\underline{i} = i_1, \ldots, i_n$, then $a[\underline{i}]$ is the tuple of terms $a_1[i_1], \ldots, a_s[i_n]$.

Format for unsafety problems formulae

Proposed format for K: $\exists^{I}$-formulae

$$\exists \underline{i}\ \phi(\underline{i}, a[\underline{i}])$$

where $\underline{i}$ is a tuple of variables of sort `INDEX` and ϕ is a quantifier-free $\Sigma_I \cup \Sigma_E$-formula.

Example

For instance, the formula

$$\exists i_1 \exists i_2.\ (i_1 \neq i_2 \wedge a[i_1] = \texttt{use} \wedge a[i_2] = \texttt{use})$$

expresses that mutual exclusion is violated.

Format for unsafety problems formulae

Proposed format for K: $\exists^I$-formulae

$$\exists \underline{i}\ \phi(\underline{i}, a[\underline{i}])$$

where $\underline{i}$ is a tuple of variables of sort `INDEX` and ϕ is a quantifier-free $\Sigma_I \cup \Sigma_E$-formula.

Example

For instance, the formula

$$\exists i_1 \exists i_2.\ (i_1 \neq i_2 \wedge a[i_1] = \texttt{use} \wedge a[i_2] = \texttt{use})$$

expresses that mutual exclusion is violated.

Format for transitions formulae

Proposed format for τ: we use disjunctions of formulae of the kind

$$\exists \underline{i} \, \exists e \left(\phi_L(e, \underline{i}, a[\underline{i}]) \wedge a' = \lambda j \, F(e, \underline{i}, a[\underline{i}], j, a[j]) \right) \tag{1}$$

where F is a case-defined function (cases are described by quantifier free formulae). The existentially quantified data variable $\exists e$ range over a sort $\texttt{ELEM}_i$ whose theory T_{E_i} has quantifier elimination - this e is needed for modeling timed systems.

Example

For instance, the formula

$$\exists i. \left(a[i] = \texttt{use} \wedge a' = \lambda j \, (\mathit{if} \ j = i \ \mathit{then} \ \texttt{idle} \ \mathit{else} \ a[j]) \right)$$

is one of the disjunctions of the transition of the 'bakery' algorithm.

Format for transitions formulae

Proposed format for τ: we use disjunctions of formulae of the kind

$$\exists \underline{i}\, \exists e \left(\phi_L(e, \underline{i}, a[\underline{i}]) \wedge a' = \lambda j\, F(e, \underline{i}, a[\underline{i}], j, a[j]) \right) \tag{1}$$

where F is a case-defined function (cases are described by quantifier free formulae). The existentially quantified data variable $\exists e$ range over a sort $\texttt{ELEM}_i$ whose theory T_{E_i} has quantifier elimination - this e is needed for modeling timed systems.

Example

For instance, the formula

$$\exists i.\ \left(a[i] = \texttt{use} \wedge a' = \lambda j\, (\mathit{if}\ j = i\ \mathit{then}\ \texttt{idle}\ \mathit{else}\ a[j]) \right)$$

is one of the disjunctions of the transition of the 'bakery' algorithm.

Format for transitions formulae

Universal quantifiers in guards

$$\exists \underline{i}\, \exists e \left(\phi_L(e, \underline{i}, a[\underline{i}]) \wedge \forall j\, \psi(e, \underline{i}, j, a[\underline{i}], a[j]) \wedge a' = \lambda j\, F(e, \underline{i}, a[\underline{i}], j, a[j]) \right) \quad (2)$$

can be eliminated by syntactic transformations by adopting the stopping failures model (also known as the *approximate model* in the model checking literature). Safety certifications for the stopping failures model are stronger than for the original model.

When universal guards are met, the stopping failure model is automatically adopted by our tool and a warning in this sense is displayed to inform the user.

Format for transitions formulae

Universal quantifiers in guards

$$\exists \underline{i}\, \exists e \left(\phi_L(e, \underline{i}, a[\underline{i}]) \wedge \forall j\, \psi(e, \underline{i}, j, a[\underline{i}], a[j]) \wedge a' = \lambda j\, F(e, \underline{i}, a[\underline{i}], j, a[j]) \right) \quad (2)$$

can be eliminated by syntactic transformations by adopting the stopping failures model (also known as the *approximate model* in the model checking literature). Safety certifications for the stopping failures model are stronger than for the original model.

When universal guards are met, the stopping failure model is automatically adopted by our tool and a warning in this sense is displayed to inform the user.

Format for transitions formulae

Universal quantifiers in guards

$$\exists \underline{i} \, \exists e \left(\phi_L(e, \underline{i}, a[\underline{i}]) \wedge \forall j \, \psi(e, \underline{i}, j, a[\underline{i}], a[j]) \wedge a' = \lambda j \, F(e, \underline{i}, a[\underline{i}], j, a[j]) \right) \quad (2)$$

can be eliminated by syntactic transformations by adopting the stopping failures model (also known as the *approximate model* in the model checking literature). Safety certifications for the stopping failures model are stronger than for the original model.

When universal guards are met, the stopping failure model is automatically adopted by our tool and a warning in this sense is displayed to inform the user.

Closure under pre-image

The following result guarantees that backward reachability can only produce $\exists^I$-formulae:

Theorem

If $H(a)$ is an $\exists^I$-formula, the formula

$$Pre(\tau, H) \;:=\; \exists a' \; (\tau(a, a') \wedge H(a'))$$

is A^E_I-equivalent to an effectively computable $\exists^I$-formula.

Proof is trivial (except for the extension with serial updates).
From the proof, it appears that the length of the existential prefix of $Pre(\tau, H)$ may grow w.r.t. that of H.

Closure under pre-image

The following result guarantees that backward reachability can only produce $\exists^I$-formulae:

Theorem

If $H(a)$ is an $\exists^I$-formula, the formula

$$Pre(\tau, H) \ := \ \exists a' \ (\tau(a, a') \wedge H(a'))$$

is A^E_I-equivalent to an effectively computable $\exists^I$-formula.

Proof is trivial (except for the extension with serial updates).
From the proof, it appears that the length of the existential prefix of $Pre(\tau, H)$ may grow w.r.t. that of H.

Closure under pre-image

The following result guarantees that backward reachability can only produce $\exists^I$-formulae:

Theorem

If $H(a)$ is an $\exists^I$-formula, the formula

$$Pre(\tau, H) \;:=\; \exists a' \; (\tau(a, a') \wedge H(a'))$$

is A^E_I-equivalent to an effectively computable $\exists^I$-formula.

Proof is trivial (except for the extension with serial updates).
From the proof, it appears that the length of the existential prefix of $Pre(\tau, H)$ may grow w.r.t. that of H.

Closure under pre-image

The following result guarantees that backward reachability can only produce $\exists^I$-formulae:

Theorem

If $H(a)$ is an $\exists^I$-formula, the formula

$$Pre(\tau, H) \;:=\; \exists a' \; (\tau(a, a') \wedge H(a'))$$

is A^E_I-equivalent to an effectively computable $\exists^I$-formula.

Proof is trivial (except for the extension with serial updates).
From the proof, it appears that the length of the existential prefix of $Pre(\tau, H)$ may grow w.r.t. that of H.

SMT problems for safety and fix-point checking

The following result guarantees effectiveness of satisfiability for safety and fix-point checking during backward reachability

Theorem

Suppose that Σ_I does not contain function symbols[a] and that $\mathcal{C}_I$ is closed under substructures. Then the formulae of the kind

$$\exists \underline{i}\, \forall \underline{j}\, \psi(\underline{i}, \underline{j}, a[\underline{i}], a[\underline{j}]) \tag{3}$$

(where ψ is a quantifier-free $\Sigma_I \cup \Sigma_E$-formula) are decidable for A_I^E-satisfiability.

[a] More generally, that T_I is locally finite.

SMT problems for safety and fix-point checking

The following result guarantees effectiveness of satisfiability for safety and fix-point checking during backward reachability

Theorem

Suppose that Σ_I does not contain function symbols[a] and that $\mathcal{C}_I$ is closed under substructures. Then the formulae of the kind

$$\exists \underline{i}\ \forall \underline{j}\ \psi(\underline{i}, \underline{j}, a[\underline{i}], a[\underline{j}]) \tag{3}$$

(where ψ is a quantifier-free $\Sigma_I \cup \Sigma_E$-formula) are decidable for A_I^E-satisfiability.

[a] More generally, that T_I is locally finite.

SMT problems for safety and fix-point checking

- The two assumptions on T_I are both necessary (undecidability arises otherwise); they concern the 'topology' of the parameterised system
- Closure under substructures means that you can 'delete' processes while maintaining the topology; e.g., deleting processes from a finite set, linear order, graph, forest, one still gets a finite set, linear order, graph, forest (but this is not true for rings)
- The proof of the above theorem is by quantifier instantiation, followed by purification and standard combination techniques

SMT problems for safety and fix-point checking

- The two assumptions on T_I are both necessary (undecidability arises otherwise); they concern the 'topology' of the parameterised system
- Closure under substructures means that you can 'delete' processes while maintaining the topology; e.g., deleting processes from a finite set, linear order, graph, forest, one still gets a finite set, linear order, graph, forest (but this is not true for rings)
- The proof of the above theorem is by quantifier instantiation, followed by purification and standard combination techniques

SMT problems for safety and fix-point checking

- The two assumptions on T_I are both necessary (undecidability arises otherwise); they concern the 'topology' of the parameterised system
- Closure under substructures means that you can 'delete' processes while maintaining the topology; e.g., deleting processes from a finite set, linear order, graph, forest, one still gets a finite set, linear order, graph, forest (but this is not true for rings)
- The proof of the above theorem is by quantifier instantiation, followed by purification and standard combination techniques

The tool MCMT

- Obvious client-server architecture
- Client generates proof obligations (SMT problems)
- Server = state-of-the-art SMT solver (invoked via API)[2]
- Crucial heuristics: primitive differentiaed format for formulae in backward search, filtering modulo enumerated datatypes, bakward and forward redundancy, ...
- Additional features: special forms of invariant synthesis and limited forms of acceleration are supported.
- Abstraction/refinement cycles implementation is ongoing work.
- The tool has been successfully tested on cache coherence, broadcast, mutual exclusion, fault tolerant, parameterized timed and also sequential problems.

[2] `Yices` is the SMT-solver employed in MCMT.

The tool MCMT

Thanks to our *fully symbolic* approach, MCMT works with very compact state descriptions:

Example

The single formula

$$\exists i_1, \ldots, i_6 \left(\begin{array}{l} pc[i_1] = loc_1 \wedge pc[i_2] = loc_2 \wedge pc[i_3] = loc_3 \wedge \\ \wedge\ pc[i_4] = loc_4 \wedge pc[i_5] = loc_5 \wedge pc[i_6] = loc_6 \wedge \\ \wedge\ i_1 < i_2 \ \wedge\ i_1 < i_3 \ \wedge\ i_1 < i_4 \ \wedge\ i_1 < i_5 \ \wedge\ i_1 < i_6 \end{array} \right)$$

represents an upset with 5! minimal states.

Example

The single formula

$$\exists i_1, i_2, i_3 \ (pc[i_1] = loc_1 \wedge pc[i_2] = loc_2 \wedge pc[i_3] = loc_3 \wedge b_1[i_1] = b_2[i_1] \wedge b_1[i_2] = b_2[i_2] \wedge b_1[i_3] = b_2[i_3])$$

(here b_1, b_2 are boolean flags) represents an upset with 2^3 minimal states.

More ...

... to be found in the comprehensive paper:

Ghilardi, S., Ranise, S.: Backward reachability of array-based systems by SMT-solving: termination and invariant synthesis. Logical Methods in Computer Science (2010).

Free download (executables, user manual, bechmarks files, ...) at

http://homes.dsi.unimi.it/~ghilardi/mcmt

Wqo theories

The key ingredient of our automated termination analysis is the following notion:

Definition

A *wqo-theory* is a universal theory whose finitely generated models are a well-quasi-order with respect to the embeddability relation.

By Higman lemma:

Example

The theory of a linear poset endowed with finitely many unary predicates is a wqo theory.

Wqo theories and termination

We fix:

- an array based system $\mathcal{S} = \langle I(a), \tau(a, a') \rangle$ based on A_E^I,
- an unsafe formula $U(a)$,
- a wqo theory W (to be seen as an abstraction of A_E^I),
- a syntactic interpretation $(-)^*$ of W into A_E^I.

Termination of backward search from U is guaranteed if we show that *formulae arising during backward search are translations* (in the sense of $(-)^*$) *of existential formulae of W*.

Wqo theories and termination

We fix:

- an array based system $\mathcal{S} = \langle I(a), \tau(a, a') \rangle$ based on A_E^I,
- an unsafe formula $U(a)$,
- a wqo theory W (to be seen as an abstraction of A_E^I),
- a syntactic interpretation $(-)^*$ of W into A_E^I.

Termination of backward search from U is guaranteed if we show that *formulae arising during backward search are translations* (in the sense of $(-)^*$) *of existential formulae of W*.

Wqo theories and termination

We fix:

- an array based system $\mathcal{S} = \langle I(a), \tau(a, a') \rangle$ based on A_E^I,
- an unsafe formula $U(a)$,
- a wqo theory W (to be seen as an abstraction of A_E^I),
- a syntactic interpretation $(-)^*$ of W into A_E^I.

Termination of backward search from U is guaranteed if we show that *formulae arising during backward search are translations* (in the sense of $(-)^*$) *of existential formulae of W*.

Wqo theories and termination

We fix:

- an array based system $\mathcal{S} = \langle I(a), \tau(a, a') \rangle$ based on A_E^I,
- an unsafe formula $U(a)$,
- a wqo theory W (to be seen as an abstraction of A_E^I),
- a syntactic interpretation $(-)^*$ of W into A_E^I.

Termination of backward search from U is guaranteed if we show that *formulae arising during backward search are translations* (in the sense of $(-)^*$) *of existential formulae of W*.

Wqo theories and termination

We fix:

- an array based system $\mathcal{S} = \langle I(a), \tau(a, a') \rangle$ based on A_E^I,
- an unsafe formula $U(a)$,
- a wqo theory W (to be seen as an abstraction of A_E^I),
- a syntactic interpretation $(-)^*$ of W into A_E^I.

Termination of backward search from U is guaranteed if we show that *formulae arising during backward search are translations* (in the sense of $(-)^*$) *of existential formulae of W*.

Wqo theories and termination

We fix:

- an array based system $\mathcal{S} = \langle I(a), \tau(a, a') \rangle$ based on A_E^I,
- an unsafe formula $U(a)$,
- a wqo theory W (to be seen as an abstraction of A_E^I),
- a syntactic interpretation $(-)^*$ of W into A_E^I.

Termination of backward search from U is guaranteed if we show that *formulae arising during backward search are translations* (in the sense of $(-)^*$) *of existential formulae of W*.

Wqo theories and termination

The above observation cannot be converted directly into an algorithm, basically because there is no bound on the length of the existential prefix of formulae that can arise in backward search. We need refined formal conditions.

Termination will be guaranteed whenever some finite sets of formulae arising from the symbolic manipulations of formulae describing *small* models are translations.

The conditions we are going to introduce are all effective and can be encoded as proof obligations that can be discharged by SMT solving.

Wqo theories and termination

The above observation cannot be converted directly into an algorithm, basically because there is no bound on the length of the existential prefix of formulae that can arise in backward search. We need refined formal conditions.

Termination will be guaranteed whenever some finite sets of formulae arising from the symbolic manipulations of formulae describing *small* models are translations.

The conditions we are going to introduce are all effective and can be encoded as proof obligations that can be discharged by SMT solving.

Wqo theories and termination

The above observation cannot be converted directly into an algorithm, basically because there is no bound on the length of the existential prefix of formulae that can arise in backward search. We need refined formal conditions.

Termination will be guaranteed whenever some finite sets of formulae arising from the symbolic manipulations of formulae describing *small* models are translations.

The conditions we are going to introduce are all effective and can be encoded as proof obligations that can be discharged by SMT solving.

Termination conditions

We give a qualitative description of our conditions for termination. The conditions involve two parameters M and N:

- M is the *maximum arity* of the predicate symbols in the signature of W (W is assumed to have a finite relational signature);
- N is the *quantifier-elimination degree* (QE-degree) of the data theory relative to the existentially quantified data variable occurring in transitions.

[In the examples we usually have $N = M = 2$.]

Termination conditions

We give a qualitative description of our conditions for termination. The conditions involve two parameters M and N:

- M is the *maximum arity* of the predicate symbols in the signature of W (W is assumed to have a finite relational signature);
- N is the *quantifier-elimination degree* (QE-degree) of the data theory relative to the existentially quantified data variable occurring in transitions.

[In the examples we usually have $N = M = 2$.]

Termination conditions

We give a qualitative description of our conditions for termination. The conditions involve two parameters M and N:

- M is the *maximum arity* of the predicate symbols in the signature of W (W is assumed to have a finite relational signature);
- N is the *quantifier-elimination degree* (QE-degree) of the data theory relative to the existentially quantified data variable occurring in transitions.

[In the examples we usually have $N = M = 2$.]

Termination conditions

We give a qualitative description of our conditions for termination. The conditions involve two parameters M and N:

- M is the *maximum arity* of the predicate symbols in the signature of W (W is assumed to have a finite relational signature);
- N is the *quantifier-elimination degree* (QE-degree) of the data theory relative to the existentially quantified data variable occurring in transitions.

[In the examples we usually have $N = M = 2$.]

QE-degree: an example

QE-degree can be a natural number or ∞. Take Fourier-Motzkin quantifier elimination for linear real arithmetic. This has QE-degree equal to 2 (wrt the set of *representative predicates* $\{=, <\}$) because:

- you can rewrite every formula as a positive DNF containing only the predicates $=, <$;
- you can eliminate a single existential quantifier from a conjunction of representative literals (i.e. positive literals containing only representative predicates) by grouping them into *pairs*, as in

$$\begin{aligned} & \exists e\,(x < e \wedge y < e \wedge e < 6) \quad \rightsquigarrow \\ \rightsquigarrow \quad & \exists e\,(x < e \wedge y < e) \wedge \exists e\,(x < e \wedge e < 6) \wedge \exists e\,(y < e \wedge e < 6) \quad \rightsquigarrow \\ \rightsquigarrow \quad & \mathit{true} \wedge x < 6 \wedge y < 6 \quad \rightsquigarrow \\ \rightsquigarrow \quad & x < 6 \wedge y < 6 \end{aligned}$$

QE-degree: an example

QE-degree can be a natural number or ∞. Take Fourier-Motzkin quantifier elimination for linear real arithmetic. This has QE-degree equal to 2 (wrt the set of *representative predicates* $\{=, <\}$) because:

- you can rewrite every formula as a positive DNF containing only the predicates $=, <$;
- you can eliminate a single existential quantifier from a conjunction of representative literals (i.e. positive literals containing only representative predicates) by grouping them into *pairs*, as in

$$\begin{aligned} & \exists e\,(x < e \wedge y < e \wedge e < 6) \quad \rightsquigarrow \\ \rightsquigarrow \quad & \exists e\,(x < e \wedge y < e) \wedge \exists e\,(x < e \wedge e < 6) \wedge \exists e\,(y < e \wedge e < 6) \quad \rightsquigarrow \\ \rightsquigarrow \quad & \mathit{true} \wedge x < 6 \wedge y < 6 \quad \rightsquigarrow \\ \rightsquigarrow \quad & x < 6 \wedge y < 6 \end{aligned}$$

QE-degree: an example

QE-degree can be a natural number or ∞. Take Fourier-Motzkin quantifier elimination for linear real arithmetic. This has QE-degree equal to 2 (wrt the set of *representative predicates* $\{=, <\}$) because:

- you can rewrite every formula as a positive DNF containing only the predicates $=, <$;
- you can eliminate a single existential quantifier from a conjunction of representative literals (i.e. positive literals containing only representative predicates) by grouping them into *pairs*, as in

$$
\begin{aligned}
& \exists e\,(x < e \wedge y < e \wedge e < 6) \quad \rightsquigarrow \\
\rightsquigarrow \quad & \exists e\,(x < e \wedge y < e) \wedge \exists e\,(x < e \wedge e < 6) \wedge \exists e\,(y < e \wedge e < 6) \quad \rightsquigarrow \\
\rightsquigarrow \quad & \mathit{true} \wedge x < 6 \wedge y < 6 \quad \rightsquigarrow \\
\rightsquigarrow \quad & x < 6 \wedge y < 6
\end{aligned}
$$

Termination conditions

Our termination conditions ask for the following to happen:

(1) the unsafe formula must be a translation;

(2) the translation of the diagram (i.e. of the 'multiplication table') of a model of W having cardinality at most M must be (in a 'smooth way') a conjunction of representative literals;

(3) case distinctions and guards in transitions must be conjunctions of representative literals;

(4) taking the pre-image (with respect to any transition and any formal case-marking) of the translation of the diagram of a model of W having cardinality at most $N * M$, one gets a formula which is equivalent to the translation of an existential formula of W.

Termination conditions

Our termination conditions ask for the following to happen:

(1) the unsafe formula must be a translation;

(2) the translation of the diagram (i.e. of the 'multiplication table') of a model of W having cardinality at most M must be (in a 'smooth way') a conjunction of representative literals;

(3) case distinctions and guards in transitions must be conjunctions of representative literals;

(4) taking the pre-image (with respect to any transition and any formal case-marking) of the translation of the diagram of a model of W having cardinality at most $N * M$, one gets a formula which is equivalent to the translation of an existential formula of W.

Termination conditions

Our termination conditions ask for the following to happen:

(1) the unsafe formula must be a translation;

(2) the translation of the diagram (i.e. of the 'multiplication table') of a model of W having cardinality at most M must be (in a 'smooth way') a conjunction of representative literals;

(3) case distinctions and guards in transitions must be conjunctions of representative literals;

(4) taking the pre-image (with respect to any transition and any formal case-marking) of the translation of the diagram of a model of W having cardinality at most $N * M$, one gets a formula which is equivalent to the translation of an existential formula of W.

Termination conditions

Our termination conditions ask for the following to happen:

(1) the unsafe formula must be a translation;

(2) the translation of the diagram (i.e. of the 'multiplication table') of a model of W having cardinality at most M must be (in a 'smooth way') a conjunction of representative literals;

(3) case distinctions and guards in transitions must be conjunctions of representative literals;

(4) taking the pre-image (with respect to any transition and any formal case-marking) of the translation of the diagram of a model of W having cardinality at most $N * M$, one gets a formula which is equivalent to the translation of an existential formula of W.

Termination conditions

Our termination conditions ask for the following to happen:

(1) the unsafe formula must be a translation;

(2) the translation of the diagram (i.e. of the 'multiplication table') of a model of W having cardinality at most M must be (in a 'smooth way') a conjunction of representative literals;

(3) case distinctions and guards in transitions must be conjunctions of representative literals;

(4) taking the pre-image (with respect to any transition and any formal case-marking) of the translation of the diagram of a model of W having cardinality at most $N * M$, one gets a formula which is equivalent to the translation of an existential formula of W.

Requirement (4) means the following:

- Take the diagram of a model of W having cardinality at most $N * M$ (e.g. suppose the model has 3 elements i_1, i_2, i_3).
- Translate the diagram and get a formula $\exists i_1\ i_2\ i_3 \psi$.
- Take a transition

$$\exists \underline{i}\, \exists e \left(\phi_L(e, \underline{i}, a[\underline{i}]) \wedge a' = \lambda j\, F(e, \underline{i}, a[\underline{i}], j, a[j]) \right) \tag{4}$$

and suppose that $F = F_1, \ldots, F_s$ is

$$\begin{aligned} F_l(j, a[j]) \ := \ & \texttt{case of} \ \{ \\ & C_1(j, \ldots) : t_{l1}(j, \ldots); \\ & \ldots \\ & C_m(j, \ldots) : t_{lm}(j, \ldots) \\ & \}, \end{aligned}$$

for $l = 1, \ldots, s$.

Requirement (4) means the following:

- Take the diagram of a model of W having cardinality at most $N * M$ (e.g. suppose the model has 3 elements i_1, i_2, i_3).
- Translate the diagram and get a formula $\exists i_1\ i_2\ i_3 \psi$.
- Take a transition

$$\exists \underline{i}\, \exists e \left(\phi_L(e, \underline{i}, a[\underline{i}]) \wedge a' = \lambda j\, F(e, \underline{i}, a[\underline{i}], j, a[j]) \right) \tag{4}$$

and suppose that $F = F_1, \ldots, F_s$ is

$$F_l(j, a[j]) := \texttt{case of } \{ \; C_1(j, \ldots) : t_{l1}(j, \ldots); \; \ldots \; C_m(j, \ldots) : t_{lm}(j, \ldots) \; \},$$

for $l = 1, \ldots, s$.

Requirement (4) means the following:

- Take the diagram of a model of W having cardinality at most $N * M$ (e.g. suppose the model has 3 elements i_1, i_2, i_3).
- Translate the diagram and get a formula $\exists i_1\ i_2\ i_3 \psi$.
- Take a transition

$$\exists \underline{i}\, \exists e \left(\phi_L(e, \underline{i}, a[\underline{i}]) \wedge a' = \lambda j\, F(e, \underline{i}, a[\underline{i}], j, a[j]) \right) \tag{4}$$

and suppose that $F = F_1, \ldots, F_s$ is

$$\begin{aligned} F_l(j, a[j]) \ := \ & \texttt{case of} \ \{ \\ & C_1(j, \ldots) : t_{l1}(j, \ldots); \\ & \ldots \\ & C_m(j, \ldots) : t_{lm}(j, \ldots) \\ & \}, \end{aligned}$$

for $l = 1, \ldots, s$.

- 'Guess' which case apply to each element of the model (e.g that case 1 applies to i_1, i_2 and case 4 to i_3).
- Make the corresponding formal update in ψ and get $\psi\sigma$ (i.e. to get $\psi\sigma$, replace the $a[i_1]$ with the $t_{*1}(i_1)$, the $a[i_2]$ with the $t_{*1}(i_2)$ and the $a[i_3]$ with the $t_{*4}(i_3)$).
- Take the conjunction of $\psi\sigma$, of the guard ϕ_L, and of the case-formulae (i.e. $C_1(i_1, \dots) \wedge C_1(i_2, \dots) \wedge C_4(i_3, \dots)$).
- Add the existential quantifier prefix $\exists e$ to the last formula and eliminate it.

Condition (4) is satisfied if the outcome is a translation, no matter which small model, transition and case-marking you used. In our example, to be a translation means to be equivalent to a formula of the kind $\exists i_1\ i_2\ i_3\ \alpha(i_1, i_2, i_3)^*$; notice that we have finitely many possibilities for α because α is quantifier-free, the tuple i_1, i_2, i_3 is fixed and the signature of W is finite and relational.

- 'Guess' which case apply to each element of the model (e.g that case 1 applies to i_1, i_2 and case 4 to i_3).
- Make the corresponding formal update in ψ and get $\psi\sigma$ (i.e. to get $\psi\sigma$, replace the $a[i_1]$ with the $t_{*1}(i_1)$, the $a[i_2]$ with the $t_{*1}(i_2)$ and the $a[i_3]$ with the $t_{*4}(i_3)$).
- Take the conjunction of $\psi\sigma$, of the guard ϕ_L, and of the case-formulae (i.e. $C_1(i_1, \dots) \wedge C_1(i_2, \dots) \wedge C_4(i_3, \dots)$).
- Add the existential quantifier prefix $\exists e$ to the last formula and eliminate it.

Condition (4) is satisfied if the outcome is a translation, no matter which small model, transition and case-marking you used. In our example, to be a translation means to be equivalent to a formula of the kind $\exists i_1\ i_2\ i_3\ \alpha(i_1, i_2, i_3)^*$; notice that we have finitely many possibilities for α because α is quantifier-free, the tuple i_1, i_2, i_3 is fixed and the signature of W is finite and relational.

- 'Guess' which case apply to each element of the model (e.g that case 1 applies to i_1, i_2 and case 4 to i_3).
- Make the corresponding formal update in ψ and get $\psi\sigma$ (i.e. to get $\psi\sigma$, replace the $a[i_1]$ with the $t_{*1}(i_1)$, the $a[i_2]$ with the $t_{*1}(i_2)$ and the $a[i_3]$ with the $t_{*4}(i_3)$).
- Take the conjunction of $\psi\sigma$, of the guard ϕ_L, and of the case-formulae (i.e. $C_1(i_1, \dots) \wedge C_1(i_2, \dots) \wedge C_4(i_3, \dots)$).
- Add the existential quantifier prefix $\exists e$ to the last formula and eliminate it.

Condition (4) is satisfied if the outcome is a translation, no matter which small model, transition and case-marking you used. In our example, to be a translation means to be equivalent to a formula of the kind $\exists i_1\ i_2\ i_3\ \alpha(i_1, i_2, i_3)^*$; notice that we have finitely many possibilities for α because α is quantifier-free, the tuple i_1, i_2, i_3 is fixed and the signature of W is finite and relational.

- 'Guess' which case apply to each element of the model (e.g that case 1 applies to i_1, i_2 and case 4 to i_3).
- Make the corresponding formal update in ψ and get $\psi\sigma$ (i.e. to get $\psi\sigma$, replace the $a[i_1]$ with the $t_{*1}(i_1)$, the $a[i_2]$ with the $t_{*1}(i_2)$ and the $a[i_3]$ with the $t_{*4}(i_3)$).
- Take the conjunction of $\psi\sigma$, of the guard ϕ_L, and of the case-formulae (i.e. $C_1(i_1, \dots) \wedge C_1(i_2, \dots) \wedge C_4(i_3, \dots)$).
- Add the existential quantifier prefix $\exists e$ to the last formula and eliminate it.

Condition (4) is satisfied if the outcome is a translation, no matter which small model, transition and case-marking you used. In our example, to be a translation means to be equivalent to a formula of the kind $\exists i_1\ i_2\ i_3\ \alpha(i_1, i_2, i_3)^*$; notice that we have finitely many possibilities for α because α is quantifier-free, the tuple i_1, i_2, i_3 is fixed and the signature of W is finite and relational.

- 'Guess' which case apply to each element of the model (e.g that case 1 applies to i_1, i_2 and case 4 to i_3).
- Make the corresponding formal update in ψ and get $\psi\sigma$ (i.e. to get $\psi\sigma$, replace the $a[i_1]$ with the $t_{*1}(i_1)$, the $a[i_2]$ with the $t_{*1}(i_2)$ and the $a[i_3]$ with the $t_{*4}(i_3)$).
- Take the conjunction of $\psi\sigma$, of the guard ϕ_L, and of the case-formulae (i.e. $C_1(i_1, \dots) \wedge C_1(i_2, \dots) \wedge C_4(i_3, \dots)$).
- Add the existential quantifier prefix $\exists e$ to the last formula and eliminate it.

Condition (4) is satisfied if the outcome is a translation, no matter which small model, transition and case-marking you used. In our example, to be a translation means to be equivalent to a formula of the kind $\exists i_1\ i_2\ i_3\ \alpha(i_1, i_2, i_3)^*$; notice that we have finitely many possibilities for α because α is quantifier-free, the tuple i_1, i_2, i_3 is fixed and the signature of W is finite and relational.

Termination conditions

Theorem

Termination is guaranteed if the above conditions (1)-(4) are satisfied.

Testing the above condition is done via SMT-solvers proof obbligations. Translations can be suggested by the user, although in principle a tool can enumerate and check them one-by-one, until a good one is found (provided one exists).

Although the models to be checked are very small, combinatorial explosion may arise if the signature of W is relatively large (simple but powerful heuristics are needed in practical cases).

Termination conditions

Theorem

Termination is guaranteed if the above conditions (1)-(4) are satisfied.

Testing the above condition is done via SMT-solvers proof obbligations. Translations can be suggested by the user, although in principle a tool can enumerate and check them one-by-one, until a good one is found (provided one exists).

Although the models to be checked are very small, combinatorial explosion may arise if the signature of W is relatively large (simple but powerful heuristics are needed in practical cases).

Termination conditions

Theorem

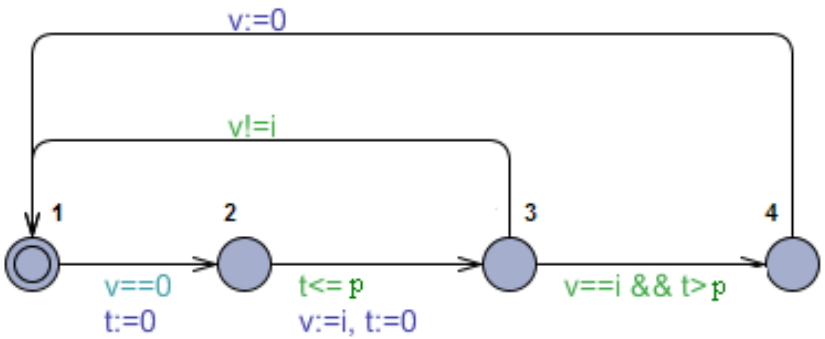
Termination is guaranteed if the above conditions (1)-(4) are satisfied.

Testing the above condition is done via SMT-solvers proof obbligations. Translations can be suggested by the user, although in principle a tool can enumerate and check them one-by-one, until a good one is found (provided one exists).

Although the models to be checked are very small, combinatorial explosion may arise if the signature of W is relatively large (simple but powerful heuristics are needed in practical cases).

Applications

The method trivially applies to broadcast protocols, lossy channels systems, etc. A sophisticated translation is required for parameterized timed automata with a single real clock: this translation is precisely the logical encoding of the relevant definition from [Abdulla-Jonsson 03]. Another example to which the above result applies is the 'parametric' and parameterised Fischer protocol shown below.



Legend: t is the local clock value, v is a shared variable, p is a symbolic positive parameter

Thanks for attention!