Algorithms for Model Checking (2IW55)

Lecture 10

Parameterised Boolean Equation Systems (2)

Background material:

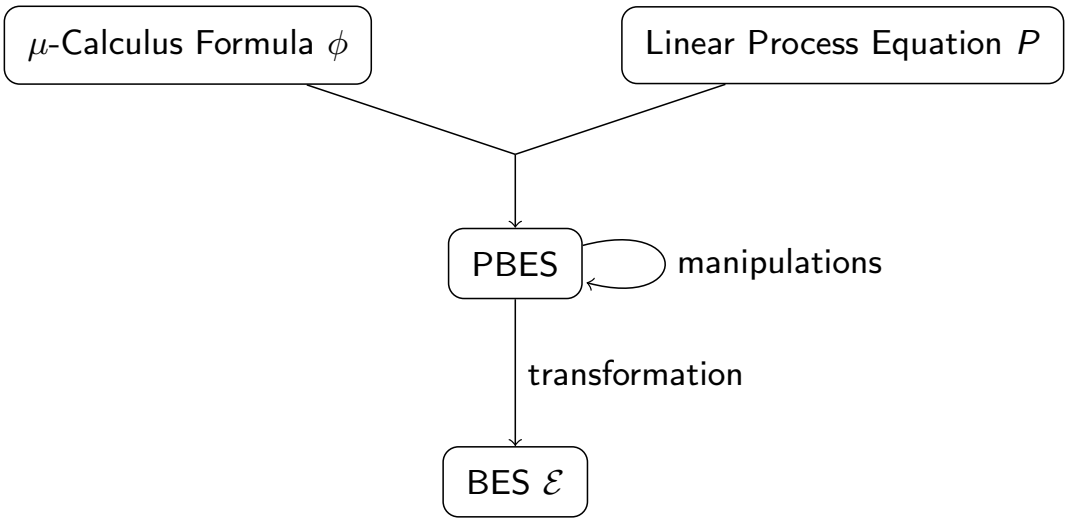
- *Verification of Reactive Systems via Instantiation of Parameterised Boolean Equation Systems*, B. Ploeger, J.W. Wesselink and T.A.C. Willemse (*I&C 2010/2011*)

- *Static Analysis Techniques for Parameterised Boolean Equation Systems*, S. Orzan, J.W. Wesselink and T.A.C. Willemse (*TACAS 2009*)

Tim Willemse
(timw@win.tue.nl)
http://www.win.tue.nl/∼timw
HG 6.76

Verification via PBESs

Verification Methodology:



Solving $\mathcal{E}$ answers $P \models \phi$

First-order Modal μ-Calculus model checking problem

- Given is a First-order Modal μ-Calculus formula $\sigma Z.\ \phi$
- Given a system described by an LPE $X(e)$

Compute whether $X(e) \models \sigma Z.\ \phi$

- Transform the model checking problem to solving a PBES
- The transformation is similar to the transformation to BES.
- Idea: for each fixed point subformula $\sigma' X.\ \psi$ of $\sigma Z.\phi$, add an equation

$$\sigma' \tilde{X}(d : D, \cdots) = RHS(\psi)$$

- The order of the equations respects the subterm ordering in $\sigma Z.\phi$

- Identify a list of data variables bound outside the scope of a fixed point formula
- Given a formula Φ and some formal variable Z

Identify Bound Data Variables

$$Par(Z, b, l) = Par(Z, X, l) = []$$

$$Par(Z, \phi \wedge \psi, l) = Par(Z, \phi \vee \psi, l) = Par(Z, \phi, l) +\!\!+ Par(Z, \psi, l)$$

$$Par(Z, \forall d{:}D.\phi, l) = Par(Z, \exists d{:}D.\phi, l) = Par(Z, \phi, [d{:}D] +\!\!+ l)$$

$$Par(Z, [\alpha]\phi, l) = Par(Z, \langle\alpha\rangle\phi, l) = Par(Z, \phi, l)$$

$$Par(Z, \sigma X.\phi, l) = \begin{cases} l & \text{if } Z = X \\ Par(Z, \phi, l) & \text{otherwise} \end{cases}$$

- Let $\psi := \sigma Z.\ \phi$
- Given LPE $X(d{:}D) = \sum_{i \le n} \sum_{e_i:D_i} c_i(d, e_i) \longrightarrow a_i(f_i(d, e_i)) \cdot X(g_i(d, e_i))$

Create Equation System Outline

$$\mathbf{E}(b) = \epsilon \qquad \mathbf{E}(Z) = \epsilon$$

$$\mathbf{E}(\phi \wedge \psi) = \mathbf{E}(\phi)\ \mathbf{E}(\psi) \qquad \mathbf{E}(\phi \vee \psi) = \mathbf{E}(\phi)\ \mathbf{E}(\psi)$$

$$\mathbf{E}(\forall d'{:}D'.\phi) = \mathbf{E}(\phi) \qquad \mathbf{E}(\exists d'{:}D'.\phi) = \mathbf{E}(\phi)$$

$$\mathbf{E}([\alpha]\phi) = \mathbf{E}(\phi) \qquad \mathbf{E}(\langle\alpha\rangle\phi) = \mathbf{E}(\phi)$$

$$\mathbf{E}(\sigma Z.\phi) = \Big(\sigma Z(d{:}D, Par(Z, \psi, [])) = \mathbf{RHS}(\phi)\Big)\ \mathbf{E}(\phi)$$

- Let $\Phi := \sigma Y.\ \phi$
- Given LPE $X(d{:}D) = \sum_{i \le n} \sum_{e_i:D_i} c_i(d, e_i) \longrightarrow a_i(f_i(d, e_i)) \cdot X(g_i(d, e_i))$

RHS:

$$\mathbf{RHS}(b) = b \qquad \mathbf{RHS}(Z) = \tilde{Z}(d, Par(Z, \Phi, []))$$

$$\mathbf{RHS}(\phi \wedge \psi) = \mathbf{RHS}(\phi)\ \wedge\ \mathbf{RHS}(\psi) \qquad \mathbf{RHS}(\phi \vee \psi) = \mathbf{RHS}(\phi)\ \vee\ \mathbf{RHS}(\psi)$$

$$\mathbf{RHS}(\forall d'{:}D'.\phi) = \forall d'{:}D'.\ \mathbf{RHS}(\phi) \qquad \mathbf{RHS}(\exists d'{:}D'.\phi) = \exists d'{:}D'.\ \mathbf{RHS}(\phi)$$

$$\mathbf{RHS}(\sigma Z.\phi) = \tilde{Z}(d, Par(Z, \Phi, []))$$

$$\mathbf{RHS}(\langle\alpha\rangle\phi) = \bigvee_{i \le n} \exists e_i{:}D_i.\ \Big(c_i(d, e_i) \wedge a_i(f_i(d, e_i))\ \text{in}\ \alpha \wedge ((\mathbf{RHS}(\phi))[d := g_i(d, e_i)])\Big)$$

$$\mathbf{RHS}([\alpha]\phi) = \bigwedge_{i \le n} \forall e_i{:}D_i.\ \Big((c_i(d, e_i) \wedge a_i(f_i(d, e_i))\ \text{in}\ \alpha) \Rightarrow ((\mathbf{RHS}(\phi))[d := g_i(d, e_i)])\Big)$$

Verification via PBESs

Matching parameterised actions with action formulae:

$$
\begin{array}{ll}
a(e) \text{ in } \mathsf{true} & = \mathsf{true} \\
a(e) \text{ in } a'(e') & = (a = a' \wedge e = e') \\
a(e) \text{ in } \neg\alpha & = \neg(a(e) \text{ in } \alpha) \\
a(e) \text{ in } (\alpha \wedge \beta) & = (a(e) \text{ in } \alpha) \ \wedge (a(e) \text{ in } \beta) \\
a(e) \text{ in } (\alpha \vee \beta) & = (a(e) \text{ in } \alpha) \ \vee (a(e) \text{ in } \beta)
\end{array}
$$

Observations:

- in yields a predicate formula
- in does not introduce predicate variables

Solving PBESs

How to solve PBESs

$$X_i(e) \stackrel{?}{=} \mathsf{true} \text{ in } \mathcal{E} := \ (\sigma_1 X_1(d_1 : D_1) = \phi_1) \cdots (\sigma_n X_n(d_n : D_n) = \phi_n)$$

Known techniques for solving/simplifying $\mathcal{E}$:

- Instantiation to BES and subsequently solve the BES
- Gauß Elimination on PBES + symbolic approximation of equations
- Using patterns
- Using under/over approximation
- Invariants

Definition (Logical Equivalence)

Let ϕ, ψ be two predicates. Then ψ is logically equivalent to ϕ, denoted $\phi \leftrightarrow \psi$ iff

$$\forall \varepsilon, \eta : \ [\phi]\eta\varepsilon = [\psi]\eta\varepsilon$$

- If $\phi \leftrightarrow \psi$, then equation $\nu X(d : D) = \phi$ has the same solution as $\nu X(d : D) = \psi$ (likewise for μ)
- Useful simplifications:
  - $\text{false} \wedge \phi \leftrightarrow \text{false}$
  - $\text{true} \vee \phi \leftrightarrow \text{true}$
  - if $d \notin \text{FV}(\phi)$, then $(\exists d : D.\ \phi) \leftrightarrow (\forall d : D.\ \phi) \leftrightarrow \phi$
  - One-point rule: $(\exists d : D. d = e \wedge \phi(d)) \leftrightarrow \phi(e)$
  - One-point rule: $(\forall d : D. d = e \Rightarrow \phi(d)) \leftrightarrow \phi(e)$
- Apply logical simplifications before applying PBES manipulations/solving techniques.

Instantiation to BES:

$$X_i(e) \stackrel{?}{=} \text{true in } \mathcal{E} := \ (\sigma_1 X_1(d_1 : D_1) = \phi_1) \cdots (\sigma_n X_n(d_n : D_n) = \phi_n)$$

- Let X_i^e be a fresh propositional variable representing instance $X_i(e)$.
- The procedure below creates a BES from $\mathcal{E}$ s.t. $X_i(e) = \text{true}$ iff $X_i^e = \text{true}$
  1. For each $X_j(e_j)$ occurring in $\text{eval}(\phi_i[d_i := e])$ create a fresh variable $X_j^{e_j}$
  2. Create an equation $\sigma_i X_i^e = \tilde{\phi}_i$, where:
     - $\overline{\phi_i} = \text{eval}(\phi_i[d_i := e])$,
     - $\tilde{\phi}_i$ is $\overline{\phi_i}$ in which every $X_j(e_j)$ is replaced by $X_j^{e_j}$
  3. Repeat step 1 and 2 for every $X_j^{e_j}$ introduced in step 1, for which there is no equation
  4. Order all equations $\sigma_i X_i^e = \ldots$ according to the ordering of $\mathcal{E}$ (ordering within a block may be arbitrary)

Solving PBESs

Gauß elimination on PBESs + Symbolic Approximation:

$X_i(e) \stackrel{?}{=}$ true in $\mathcal{E} := \quad (\sigma_1 X_1(d_1 : D_1) = \phi_1) \cdots (\sigma_n X_n(d_n : D_n) = \phi_n)$

- Local solution: eliminate X in its defining equation:

$$\mathcal{E}_0\ (\sigma X(d{:}D) = \phi)\ \mathcal{E}_1 \text{ becomes } \mathcal{E}_0\ (\sigma X(d{:}D) = X^\omega)\ \mathcal{E}_1$$

  - X^ω can be found by symbolic approximation:
  - $X^0 =$ false if $\sigma = \mu$, else $X^0 =$ true
  - $X^{n+1} = \phi[X := X^n]$
  - X^ω may require transfinite approximation; else $X^\omega = X^n$ for $X^n \leftrightarrow X^{n+1}$
- Substitute solved equations (i.e. not containing predicate variables) forward:

$$\begin{array}{rl} & \mathcal{E}_0\ (\sigma_1 X_1(d_1{:}D_1) = \phi_1)\ \mathcal{E}_1\ (\sigma_2 X_2(d_2{:}D_2) = \phi_2)\ \mathcal{E}_2 \\ \text{becomes:} & \mathcal{E}_0\ (\sigma_1 X_1(d_1{:}D_1) = \phi_1)\ \mathcal{E}_1\ (\sigma_2 X_2(d_2{:}D_2) = \phi_2[X_1 := \phi_1])\ \mathcal{E}_2 \end{array}$$

- Substitute definition backwards:

$$\begin{array}{rl} & \mathcal{E}_0\ (\sigma_1 X_1(d_1{:}D_1) = \phi_1)\ \mathcal{E}_1\ (\sigma_2 X_2(d_2{:}D_2) = \phi_2)\ \mathcal{E}_2 \\ \text{becomes:} & \mathcal{E}_0\ (\sigma_1 X_1(d_1{:}D_1) = \phi_1[X_2 := \phi_2])\ \mathcal{E}_1\ (\sigma_2 X_2(d_2{:}D_2) = \phi_2)\ \mathcal{E}_2 \end{array}$$

Solving PBESs

Gauß Elimination terminates; symbolic approximation may not terminate

- Due to infinite data types, a transfinite approximation may be needed
- Evaluating predicates may be impossible: $\exists k, l, m : Nat.x^k + y^l = z^m$
- Theorem proving technology may be added in symbolic approximation

Instantiation may not terminate

- Consider the below formula that asserts that an infinite a-path is possible:

  $$\nu X.\langle a \rangle X$$

- Consider the infinite state process M:

  $$M(n : Nat) = \text{true} \longrightarrow a \cdot M(n+1)$$

- Resulting PBES:

  $$\nu \tilde{X}(n : Nat) = \text{true} \wedge \tilde{X}(n+1)$$

- Problem: always one new variable must be further investigated:
  X^0 depends on X^1 depends on X^2 depends on...

Definition (Simple Formula)

A simple formula is a formula not containing predicate variables

Observations:

1. Consider the equation $\nu X(n : Nat) = \text{true} \wedge X(n+1)$
   - X has solution true (check!)
   - Consider formal parameter n:
   - It does not affect the value of the simple subformula true
   - It appears to be redundant for the solution to X
2. Consider the equation $\nu X(n : Nat, m : Nat) = n \leq 5 \wedge X(n+m, m)$
   - X has solution $n \leq 5 \wedge m = 0$ (check!)
   - Consider formal parameter m:
   - It does not affect the value of the simple formula $n \leq 5$
   - Via a single recursion through X, it does affect the value of $n \leq 5$
   - It appears to become significant for the solution to X

- Identify all obvious significant formal parameters . . . . . sig
- Identify the dependencies . . . . . dep

$$
\begin{array}{ll}
\mathsf{sig}(b) & = \mathsf{FV}(b) \\
\mathsf{sig}(X(e)) & = \emptyset \\
\mathsf{sig}(\phi \wedge \psi) & = \mathsf{sig}(\phi) \cup \mathsf{sig}(\psi) \\
\mathsf{sig}(\phi \vee \psi) & = \mathsf{sig}(\phi) \cup \mathsf{sig}(\psi) \\
\mathsf{sig}(\forall d{:}D.\ \phi) & = \mathsf{sig}(\phi) \setminus \{d\} \\
\mathsf{sig}(\exists d{:}D.\ \phi) & = \mathsf{sig}(\phi) \setminus \{d\}
\end{array}
\qquad
\begin{array}{ll}
\mathsf{dep}(b) & = \emptyset \\
\mathsf{dep}(X(e)) & = \{X(e)\} \\
\mathsf{dep}(\phi \wedge \psi) & = \mathsf{dep}(\phi) \cup \mathsf{dep}(\psi) \\
\mathsf{dep}(\phi \vee \psi) & = \mathsf{dep}(\phi) \cup \mathsf{dep}(\psi) \\
\mathsf{dep}(\forall d{:}D.\ \phi) & = \mathsf{dep}(\phi) \\
\mathsf{dep}(\exists d{:}D.\ \phi) & = \mathsf{dep}(\phi)
\end{array}
$$

Examples:

- $\mathsf{sig}(\mathsf{true} \wedge X(n+1)) = \emptyset$, $\mathsf{sig}(n \leq 5 \wedge X(n+m, m)) = \{n\}$
- $\mathsf{dep}(\mathsf{true} \wedge X(n+1)) = \{X(n+1)\}$, $\mathsf{dep}(n \leq 5 \wedge X(n+m, m)) = \{X(n+m, m)\}$

Assume the following PBES:

$$\mathcal{E} := \quad (\sigma_1 X_1(d_1 : D_1) = \phi_1) \cdots (\sigma_n X_n(d_n : D_n) = \phi_n)$$

- $\mathsf{arity}(X_i)$: the length of vector d_i
- $d_i[j]$ denotes the j-th element of vector d_i

- Construct a marked influence graph $G(\mathcal{E}) = \langle V, \longrightarrow, M \rangle$:
- $V = \{(X_i, j) \mid 1 \leq j \leq \mathsf{arity}(X_i)\}$ is the set of vertices
- $(X_i, k) \longrightarrow (X_j, l)$ iff for some expression e: $X_j(e) \in \mathsf{dep}(\phi_i)$ and $d_i[k] \in \mathsf{FV}(e[l])$
- $M = \{(X_i, j) \mid 1 \leq i \leq n \text{ and } d_i[j] \in \mathsf{sig}(\phi_i)\}$ is the marking

Definition (Positive redundant parameters)

Given a Marked Influence Graph $G(\mathcal{E}) = \langle V, \longrightarrow, M \rangle$.
The set of positive redundant parameters of $\mathcal{E}$ is:

$$\mathcal{R} = \{d_i[j] \mid (X_i, j) \not\longrightarrow^* (X_k, l) \text{ and } (X_k, l) \in M\}$$

- Computing the set $\mathcal{R}$ requires $\mathcal{O}(| \longrightarrow |)$ steps at most
- $\mathcal{R}$ can be computed using a standard least fixed point computation, a depth-first search or a breadth-first search.

Given closed equation system $\mathcal{E}$ with no unbound data variables

Procedure for eliminating redundant parameters in $\mathcal{E}$

1. Step 1 (compute redundant parameters)
   - 1.1 Construct Marked Influence Graph of $\mathcal{E}$
   - 1.2 Compute the set $\mathcal{R}$ of positive redundant parameters of $\mathcal{E}$
2. Step 2 (remove redundant parameters): for every equation $\sigma_i X_i(d_i{:}D_i) = \phi_i$ in $\mathcal{E}$:
   - 2.1 remove parameter $d_i[j]$ from $X_i(d_i{:}D_i)$ iff $d_i[j] \in \mathcal{R}$
   - 2.2 remove expression $e[j]$ from an occurrence $X_k(e)$ in ϕ_i iff $d_k[j] \in \mathcal{R}$

Theorem (Redundancy)

The modified equation system $\mathcal{E}$ has the "same" solution as $\mathcal{E}$, i.e., the solution of a variable X does not depend on the parameters that have been identified as positively redundant.