Algorithms for Model Checking (2IW55)

Lecture 15

Parameterised Boolean Equation Systems

Tim Willemse
(timw@win.tue.nl)
http://www.win.tue.nl/~timw
HG 6.81

Outline

Transforming Satisfiability to Solving PBESs

Transformation of the First-order Modal μ-Calculus to PBES

- Given is a First-order Modal μ-Calculus formula $\psi := \sigma Z(d_f : D_f := e_f).\ \phi$
- Given a system described by an LPE $X(e)$ over Act:

$$X(d : D) = \sum_{i \leq n} \sum_{e_i : D_i} c_i(d, e_i) \longrightarrow a_i(f_i(d, e_i)) \cdot X(g_i(d, e_i))$$

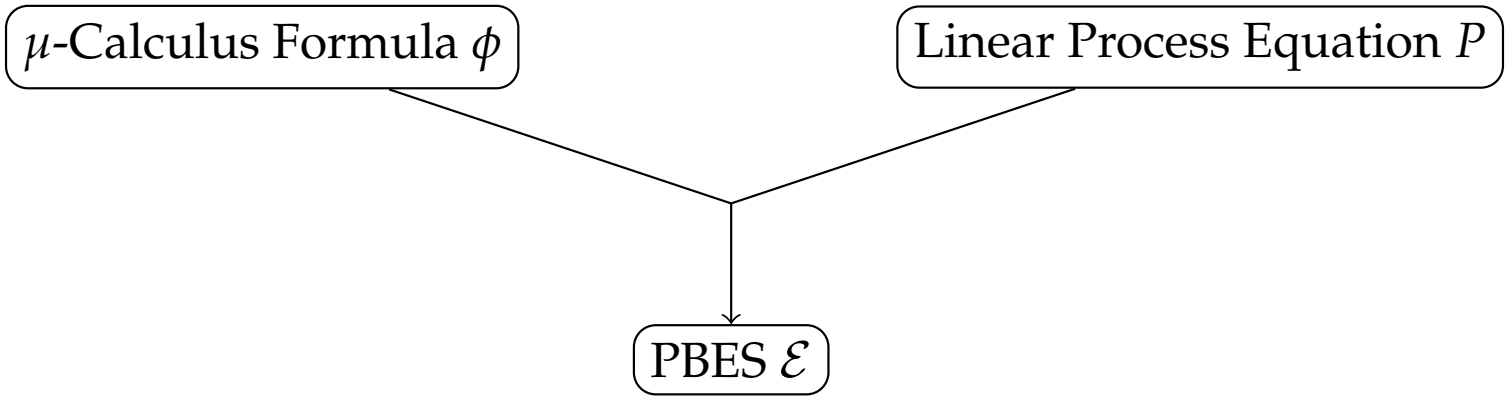
- The transformation of the model checking problem for the First-order Modal μ-Calculus is similar to the transformation to BES:
  - For each subformula $\sigma Z(d_f : D_f := e_f).\phi$, we add an equation

$$(\sigma \tilde{Z}(d : D, d_f : D_f, Par(Z, \psi)) = RHS(\phi))$$

  - $Par(Z, \psi)$ contains the smallest vector of variables (+ their sorts) that may occur free within the scope of $\sigma Z \ldots$ in the original formula ψ
  - The order of the equations respects the subterm ordering in ψ

Transforming Satisfiability to Solving PBESs

Methodology:



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

Transforming Satisfiability to Solving PBESs

- Given is a First-order Modal μ-Calculus formula $\psi := \sigma Z(d_f : D_f := e_f).\ \phi$
- Given a system described by an LPE $X(e)$ over Act:

$$X(d : D) = \sum_{i \leq n} \sum_{e_i : D_i} c_i(d, e_i) \longrightarrow a_i(f_i(d, e_i)) \cdot X(g_i(d, e_i))$$

- Operator $\mathbf{E}(\psi)$ breaks down the structure of ψ and generates equations

$$\mathbf{E}(\phi) = \epsilon \ldots\ldots\ldots\ldots\ldots \text{for } \phi \in \{\mathsf{true}, \mathsf{false}, b, Z(e)\}$$

$$\mathbf{E}(\phi_1 \Box \phi_2) = \mathbf{E}(\phi_1)\ \mathbf{E}(\phi_2) \ldots\ldots\ldots\ldots\ldots \text{for } \Box \in \{\vee, \wedge\}$$

$$\mathbf{E}(\mathbf{Q} d : D.\phi) = \mathbf{E}(\phi) \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{for } \mathbf{Q} \in \{\exists, \forall\}$$

$$\mathbf{E}(\mathbf{M}_\alpha \phi) = \mathbf{E}(\phi) \ldots\ldots\ldots\ldots\ldots\ldots \text{for } \mathbf{M}_\alpha \in \{[\alpha], \langle \alpha \rangle\}$$

$$\mathbf{E}((\sigma Z(d_f : D_f := e_f).\phi)) = (\sigma Z(d : D, d_f : D_f, Par(Z, \psi)) = \mathbf{RHS}(\phi))\ \mathbf{E}(\phi)$$

Transforming Satisfiability to Solving PBESs

Example

The one-place buffer system described by process B:

$$\begin{aligned} B(b : Bool, n : Nat) \quad &= \quad \sum_{m:Nat} \; b \longrightarrow r(m) \cdot B(\mathsf{false}, m) \\ &+ \quad \neg b \longrightarrow s(n) \cdot B(\mathsf{true}, n) \end{aligned}$$

Property ψ:if the input stream is constant, so is the output stream. Formally:

$$\forall k : Nat.\ (\nu X.(\forall l : Nat.\ [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)))$$

Transform ψ to a formula χ that starts with a dummy fixed point:

$$\nu A.\ \forall k : Nat.\ (\nu X.(\forall l : Nat.\ [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)))$$

We have: $Par(A, \chi) = []$ and $Par(X, \chi) = [k : Nat]$

Transforming Satisfiability to Solving PBESs

Example

Applying operator $\mathbf{E}$ on χ given B:

$$
\begin{aligned}
& \mathbf{E}(\chi) \\
= \quad & \mathbf{E}(\underline{\nu A.}\ \chi_1) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\chi_1))\ \mathbf{E}(\chi_1) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\chi_1))\ \mathbf{E}(\forall k : Nat.\ \chi_2) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\chi_1))\ \mathbf{E}(\forall k : Nat.\ \chi_2) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\chi_1))\ \mathbf{E}(\nu X.\chi_3) \\
= \quad & (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\chi_1)) \\
& (\nu \tilde{X}(b : Bool, n : Nat, k : Nat) = \mathbf{RHS}(\chi_3))\ \mathbf{E}(\chi_3) \\
= \quad & \ldots \\
& (\nu \tilde{A}(b : Bool, n : Nat) = \mathbf{RHS}(\chi_1)) \\
& (\nu \tilde{X}(b : Bool, n : Nat, k : Nat) = \mathbf{RHS}(\chi_3))
\end{aligned}
$$

So, $\mathbf{E}(\chi)$ yields two equations.

Transforming Satisfiability to Solving PBESs

- Given is a First-order Modal μ-Calculus formula $\psi := \sigma Z(d_f : D_f := e_f).\ \phi$
- Given a system described by an LPE $X(e)$ over Act:

$$X(d : D) = \sum_{i \leq n} \sum_{e_i : D_i} c_i(d, e_i) \longrightarrow a_i(f_i(d, e_i)) \cdot X(g_i(d, e_i))$$

- Operator $\mathbf{RHS}(\psi)$ breaks down the structure of ψ and generates predicates

$$\mathbf{RHS}(\phi) = \phi \ldots\ldots\ldots\ldots\ldots\ldots \text{for } \phi \in \{\mathsf{true}, \mathsf{false}, b\}$$

$$\mathbf{RHS}(Z(e)) = \tilde{Z}(d, e, Par(Z, \psi))$$

$$\mathbf{RHS}(\phi_1 \Box \phi_2) = \mathbf{RHS}(\phi_1) \Box \mathbf{RHS}(\phi_2) \ldots\ldots\ldots \text{for } \Box \in \{\vee, \wedge\}$$

$$\mathbf{RHS}(\mathbf{Q}d : D.\phi) = \mathbf{Q}d : D.\ \mathbf{RHS}(\phi) \ldots\ldots\ldots\ldots\ldots \text{for } \mathbf{Q} \in \{\exists, \forall\}$$

$$\mathbf{RHS}((\sigma Z(d_f : D_f := e_f).\phi)) = \tilde{Z}(d, e_f, Par(Z, \psi))$$

Transforming Satisfiability to Solving PBESs

- Given is a First-order Modal μ-Calculus formula $\psi := \sigma Z(d_f : D_f := e_f).\ \phi$
- Given a system described by an LPE $X(e)$ over Act:

$$X(d:D) = \sum_{i \le n} \sum_{e_i : D_i} c_i(d, e_i) \longrightarrow a_i(f_i(d, e_i)) \cdot X(g_i(d, e_i))$$

- Operator $\mathbf{RHS}(\psi)$ breaks down the structure of ψ and generates predicates

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

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

Transforming Satisfiability to Solving PBESs

Example

Given the buffer process B of the previous example, and subformula χ_3 of χ:
$(\forall l : Nat.\ [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X))$

$$
\begin{aligned}
& \mathbf{RHS}(\forall l : Nat.\ [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)) \\
= \quad & \forall l : Nat.\ \mathbf{RHS}([r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)) \\
= \quad & \forall l : Nat.\ (\mathbf{RHS}([r(l)](l = k \Rightarrow X)) \wedge \mathbf{RHS}([s(l)](l = k \wedge X)))
\end{aligned}
$$

For constructing $\mathbf{RHS}([r(l)](l = k \Rightarrow X))$ we have to use process B:

$$
\begin{aligned}
& \mathbf{RHS}(([r(l)](l = k \Rightarrow X))) \\
= \quad & (\forall m : Nat.\ (b \wedge \mathsf{match}(r(m), r(l))) \Rightarrow \mathbf{RHS}(l = k \Rightarrow X)[b := \mathsf{false}, n := m]) \\
\wedge \quad & ((\neg b \wedge \mathsf{match}(s(n), r(l))) \Rightarrow \mathbf{RHS}(l = k \Rightarrow X)[b := \mathsf{true}, n := n]) \\
= \quad & (\forall m : Nat.\ (b \wedge \mathsf{match}(r(m), r(l))) \Rightarrow (l = k \Rightarrow \tilde{X}(\mathsf{false}, m, k))) \\
\wedge \quad & ((\neg b \wedge \mathsf{match}(s(n), r(l))) \Rightarrow (l = k \Rightarrow \tilde{X}(\mathsf{true}, n, k)))
\end{aligned}
$$

Transforming Satisfiability to Solving PBESs

Matching parameterised actions with action formulae can also be transformed to a predicate

$$
\begin{array}{ll}
\mathsf{match}(a_i(d_{a_i}), \mathsf{true}) & = \mathsf{true} \\
\mathsf{match}(a_i(d_{a_i}), b) & = b \\
\mathsf{match}(a_i(d_{a_i}), a(e)) & = a \approx a_i \wedge d_{a_i} = e \\
\mathsf{match}(a_i(d_{a_i}), \neg\alpha) & = \neg\mathsf{match}(a_i(d_{a_i}), \alpha) \\
\mathsf{match}(a_i(d_{a_i}), \alpha_1 \wedge \alpha_2) & = \mathsf{match}(a_i(d_{a_i}), \alpha_1) \wedge \mathsf{match}(a_i(d_{a_i}), \alpha_2) \\
\mathsf{match}(a_i(d_{a_i}), \forall d : D.\alpha) & = \forall d : D.\mathsf{match}(a_i(d_{a_i}), \alpha)
\end{array}
$$

- $\mathsf{match}(a_i(d_{a_i}))$ can always be brought into Positive Normal Form
- Hence, $\mathsf{match}(a_i(d_{a_i}))$ is a predicate formula
- $\mathsf{match}(a_i(d_{a_i}))$ does not introduce predicate variables
- $\mathsf{match}(a_i(d_{a_i}))$ will not cause problems on the left-hand side of an implication

Transforming Satisfiability to Solving PBESs

Example

- The expression $\mathsf{match}(r(m), r(l))$ yields $r = r \wedge m = l$, which simplifies to $m = l$
- The expression $\mathsf{match}(s(n), r(l))$ yields $s = r \wedge n = l$, which simplifies to false
- An expression $\mathsf{match}(s(n), \forall l : Nat.s(l))$ yields: $\forall l : Nat.\ \mathsf{match}(s(n), s(l))$, which yields $\forall l : Nat.\ (s = s \wedge n = l)$, which simplifies to $\forall l : Nat.\ n = l$, which further simplifies to false

Example

Buffer system and constant stream revisited

$$\begin{aligned} B(b : Bool, n : Nat) \quad &= \quad \sum_{m:Nat} \; b \longrightarrow r(m) \cdot B(\mathsf{false}, m) \\ &+ \quad \neg b \longrightarrow s(n) \cdot B(\mathsf{true}, n) \end{aligned}$$

Property ψ: $\forall k : Nat.\ (\nu X.(\forall l : Nat.\ [r(l)](l = k \Rightarrow X) \wedge [s(l)](l = k \wedge X)))$

Result after translation to PBES $\mathcal{E}$ (and minor simplifications):

$$\begin{aligned} &(\nu \tilde{A}(b : Bool, n : Nat) = \forall k : Nat.\tilde{X}(b, n, k)) \\ &(\nu \tilde{X}(b : Bool, n : Nat, k : Nat) = \\ &\qquad \forall l : Nat.\ ((\forall m : Nat.\ (b \wedge m = l) \Rightarrow (l = k \Rightarrow \tilde{X}(\mathsf{false}, m, k))) \\ &\wedge ((\neg b \wedge n = l) \Rightarrow (l = k \wedge \tilde{X}(\mathsf{true}, n, k))))) \end{aligned}$$

For all $b : Bool$ and $n : Nat$, we have: $B(b, n) \models \psi$ iff $([\mathcal{E}]\theta\varepsilon)(\tilde{A})(b, n) = \mathsf{true}$

Outline

Solving PBESs

How to solve PBESs

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

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

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

Solving PBESs

Instantiation to BES:

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

- Let X_i^e be a fresh propositional variable representing $X_i(e)$.
- 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 $\text{EVAL}(\phi_i[d_i := e])$ add a fresh variable $X_j^{e_j}$
  2. Creating an equation $\sigma_i X_i^e = \text{EVAL}(\phi_i[d_i := e])$ with $X_j(e_j)$ replaced by $X_j^{e_j}$
  3. Repeat step 1 and 2 for every $X_j^{e_j}$ introduced in step 1
  4. Order all equations $\sigma_i X_i^e = \ldots$ according to the ordering of $\mathcal{E}$

Solving PBESs

Example

PBES: $(\nu X(n : Nat) = n \leq 2 \wedge Y(n))\ (\mu Y(n : Nat) = \mathsf{odd}(n) \vee X(n+1))$
Instantiation starting at e.g. $X(0)$ ..... introduce X^0

1. $Y(0)$ occurs in $\text{EVAL}((n \leq 2 \wedge Y(n))[n := 0])$ ..... introduce Y^0
2. Introduce $\nu X^0 = \text{EVAL}(0 \leq 2 \wedge Y^0)$ ..... $\nu X^0 = Y^0$
3. $X(1)$ occurs in $\text{EVAL}((\mathsf{odd}(n) \vee X(n+1))[n := 0])$ ..... introduce X^1
4. Introduce $\mu Y^0 = \text{EVAL}(\mathsf{odd}(0) \vee X^1)$ ..... $\mu Y^0 = X^1$
5. $Y(1)$ occurs in $\text{EVAL}((n \leq 2 \wedge Y(n))[n := 1])$ ..... introduce Y^1
6. Introduce $\nu X^1 = \text{EVAL}(1 \leq 2 \wedge Y^1)$ ..... $\nu X^1 = Y^1$
7. no variable occurs in $\text{EVAL}((\mathsf{odd}(n) \vee X(n+1))[n := 1])$ ..... end
8. Introduce $\mu Y^1 = \text{EVAL}(\mathsf{odd}(1) \vee X^2)$ ..... $\mu Y^1 = \mathsf{true}$
9. Order equations: first X^i, then Y^j
10. Resulting BES: $(\nu X^0 = Y^0)\ (\nu X^1 = Y^1)\ (\mu Y_0 = X^1)\ (\mu Y_1 = \mathsf{true})$

Solving PBESs

Instantiation may not terminate

- Always at least one new variable must be further investigated
- Evaluating predicates may be impossible: $\exists k, l, m : Nat . x^k + y^l = z^m$
- The number of BES equations may be extremely large
- Tool implementing this strategy: pbes2bool in mCRL2
  - still experimental!
  - "genuine" infinite data sorts: Nat, Pos, Int, List,...
- CADP: similar technology, generates BES immediately on-the-fly
  - stable
  - no real infinite data sorts

Solving PBESs

Gauß elimination on PBES + Symbolic Approximation:

- 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 symbolical approximation:
  - $X^0 = \mathsf{false}$ if $\sigma = \mu$, else $X^0 = \mathsf{true}$
  - $X^{n+1} = \phi[X := X^n]$
  - X^ω may require transfinite approximation; else $X^\omega = X^n$ for $X^n = X^{n+1}$
- Substitute closed 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

Example

PBES: $(\nu X(n : Nat) = n \leq 2 \wedge Y(n))\ (\mu Y(n : Nat) = \mathsf{odd}(n) \vee X(n+1))$

1. Eliminate Y from $(\mu Y(n : Nat) = \mathsf{odd}(n) \vee X(n+1))$ ..... done
2. Substitute definition of Y backwards:

$$\begin{array}{rl} & (\nu X(n : Nat) = n \leq 2 \wedge Y(n)) \\ \text{becomes} & (\nu X(n : Nat) = n \leq 2 \wedge (\mathsf{odd}(n) \vee X(n+1))) \end{array}$$

3. Eliminate X from $(\nu X(n : Nat) = n \leq 2 \wedge (\mathsf{odd}(n) \vee X(n+1)))$:

$$\begin{array}{lll} X^0 & \equiv \mathsf{true} & \\ X^1 & \equiv n \leq 2 \wedge (\mathsf{odd}(n) \vee \mathsf{true}) & \equiv n \leq 2 \\ X^2 & \equiv n \leq 2 \wedge (\mathsf{odd}(n) \vee n+1 \leq 2) & \equiv n \leq 2 \wedge (\mathsf{odd}(n) \vee n \leq 1) \equiv n \leq 1 \\ X^3 & \equiv n \leq 2 \wedge (\mathsf{odd}(n) \vee n+1 \leq 1) & \equiv n \leq 2 \wedge (\mathsf{odd}(n) \vee n = 0) \equiv n \leq 1 \end{array}$$

So, solution to X is $n \leq 1$

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
- Tools implementing this strategy: mucheck in μCRL, experimental: pbessolve in mCRL2
- Instantiation and Gauß Elimination+symbolic approximation are sometimes complementary

Outline

Exercise

Consider the lossy channel system described by the following LPE:

$$\begin{aligned} C(b : Bool, m : M) \quad = \quad & \sum_{k:M} b \longrightarrow r(k) \cdot C(\mathsf{false}, k) \\ + \quad & \neg b \longrightarrow s(m) \cdot C(\mathsf{true}, m) \\ + \quad & \neg b \longrightarrow l \cdot C(\mathsf{true}, m) \end{aligned}$$

Action r stands for reading, s stands for sending and l stands for losing a message.

1. $\nu X.([\mathsf{true}]X \wedge (\mu Y.[\neg(\exists m : M.s(m))]Y \wedge \langle \mathsf{true} \rangle \mathsf{true}))$
2. $\nu X.\mu Y.\nu Z.[\exists m{:}M.s(m)]X \wedge ([\exists m{:}M.s(m)]\mathsf{false} \vee [\neg(\exists m{:}M.s(m))]Y) \wedge [\neg(\exists m{:}M.s(m))]Z$

Questions:

- Translate both formulae to PBESs given process $C(\mathsf{true}, m_0)$
- Use instantiation to compute BESs when $M = Bool$, and solve the BES ($m_0 = \mathsf{true}$)
- Use symbolic approximation when $M = Nat$ ($m_0 = 0$)