Lecture 1:
Verification of Concurrent Programs
Part 1: Decidability and Complexity Results

Ahmed Bouajjani

LIAFA, University Paris Diderot – Paris 7

VTSA, MPI-Saarbrücken, September 2012
Outline of the lectures

- **Lecture 1: Concurrent programs: Decidability and complexity Results**
  - Basic models
  - Limits of the decidability of the reachability problem
  - Classes of programs/models with a decidable state reachability problem

- **Lecture 2: Concurrent programs: Under-approximate analysis**
  - Bounded analysis for concurrent programs
  - Decidability and complexity issues
  - Compositional reduction to state reachability in *sequential programs*

- **Lecture 3: Weak memory models: State reachability problem**
  - Weaker models than Sequential Consistency
  - (Un)Decidability and complexity of the state reachability problem
  - Efficient under-approximate analysis: Reduction to SC state reachability

- **Lecture 4: Weak memory models: Robustness against a WMM**
  - Check that all behaviors are still sequentially consistent
  - Decidability and complexity
  - Reduction to SC state reachability
Concurrent Programs

- Parallel threads (with/without procedure calls)
- Static/Dynamic number of threads
- Communication
  - Shared memory
    - Notion of action atomicity
    - Actions by a same threads are executed in the same order (Sequential Consistency)
    - Actions by different threads are interleaved non-deterministically
  - Message passing
    - Channels (queues)
    - Unordered/FIFO ...
    - Perfect/Lossy
- We assume finite data domain (e.g., booleans).
Finite number of threads + Shared variables

- Fixed number of threads
- Iterative processes (no recursive procedure calls)
- Finite number of variables

\[ \text{A variable has a finite number of possible values} \implies \text{Finite product of finite-state systems (threads + variables)} \implies \text{Decidable} \]

Reachability is decidable, and PSPACE-complete. [Kozen, FOCS'77]
Finite number of threads + Shared variables

- Fixed number of threads
- Iterative processes (no recursive procedure calls)
- Finite number of variables
- A variable has a finite number of possible values
- $\Rightarrow$ Finite product of finite-state systems (threads + variables)
- $\Rightarrow$ Decidable
Finite number of threads + Shared variables

- Fixed number of threads
- Iterative processes (no recursive procedure calls)
- Finite number of variables
- A variable has a finite number of possible values
  - $\Rightarrow$ Finite product of finite-state systems (threads + variables)
  - $\Rightarrow$ Decidable
- Product grows exponentially in \# threads and \# variables.
- Reachability is decidable, and PSPACE-complete.
  [Kozen, FOCS’77]
Finite number of threads + bounded queues

- Fixed number of threads
- Iterative processes (no recursive procedure calls)
- Bounded channels
Finite number of threads + bounded queues

- Fixed number of threads
- Iterative processes (no recursive procedure calls)
- Bounded channels
  - \(\Rightarrow\) Finite number of possible channel contents
  - \(\Rightarrow\) Finite product of finite-state systems (threads + channels)
  - \(\Rightarrow\) Decidable
Finite number of threads + bounded queues

- Fixed number of threads
- Iterative processes (no recursive procedure calls)
- Bounded channels
- ⇒ Finite number of possible channel contents
- ⇒ Finite product of finite-state systems (threads + channels)
- ⇒ Decidable
- Product grows exponentially in # threads and size of channels.
- Reachability is decidable, and PSPACE-complete.
Facing the state-space explosion

- Partial order techniques
  - Independent actions $\Rightarrow$ commutable actions $\Rightarrow$ many interleavings
  - Explore representatives up to independent actions commutations
    Godefroid, Wolper, Peled, Holzma, Valmari, ...

- Symbolic techniques
  - Compact representations of sets of states + fixpoint calculations
  - Bounded model checking + SAT solvers
    Clarke, McMillan, Somenzi, Biere, Cimatti, ...
Beyond the finite-state case

- Unbounded (parametric/dynamic) number of threads
  - Undecidable in general if threads IDs are allowed
  - ⇒ Anonymous threads

- Unbounded channels
  - Undecidable in general in case of FIFO queues
  - ⇒ Unordered queues (multisets), lossy queues
Programs with Dynamic Creation of Threads

- Finite number of variables
- Finite data domain
- Threads are anonymous (no way to refer to identities)

Safety is reducible to state reachability in VASS / Coverability in PN
Programs with Dynamic Creation of Threads

- Finite number of variables
- Finite data domain
- ⇒ Threads are anonymous (no way to refer to identities)
- Iterative processes (no recursive procedure calls)
- ⇒ Counting abstraction
  - Finite number of possible local states $\ell_1, \ldots, \ell_m$
  - Count how many threads are in a given local state

Safety is reducible to state reachability in VASS / Coverability in PN
Programs with Dynamic Creation of Threads

- Finite number of variables
- Finite data domain
- \( \Rightarrow \) Threads are anonymous (no way to refer to identities)
- Iterative processes (no recursive procedure calls)
- \( \Rightarrow \) Counting abstraction
  - Finite number of possible local states \( \ell_1, \ldots, \ell_m \)
  - Count how many threads are in a given local state
- Safety is reducible to state reachability in VASS / Coverability in PN
Vector Addition Systems with States

- Finite state machine + finite number of counter $C = \{c_1, \ldots, c_n\}$.
- Operations: (No test to zero)
  - $c_i := c_i + 1$
  - $c_i > 0 \rightarrow c_i := c_i - 1$
- Configuration: $(q, V)$ where $q$ is a control state and $V \in \mathbb{N}^n$
- Initial configuration: $(q_0, 0)$ where $0 = 0^n$.
- Transition relation:
  \[
  (q_1, V_1) \xrightarrow{op} (q_2, V_2) \quad \text{iff}
  \]
  - $op = \text{“}c_i := c_i + 1\text{“}$, and $V_2 = V_1[c_i \leftarrow (V_1(c_i) + 1)]$
  - $op = \text{“}c_i > 0 \rightarrow c_i := c_i - 1\text{“}$, and $(V_1(c_i) > 0 \text{ and } V_2 = V_1[c_i \leftarrow (V_1(c_i) - 1)])$
From Multithreaded Programs to VASS

- Associate a control state with each valuation of the globals
- Associate a counter with each valuation of thread locals
- A statement moving globals from \( g \) to \( g' \) and locals from \( \ell \) to \( \ell' \):
  \[
  g \xrightarrow{\begin{array}{c}
c_{\ell} > 0/c_{\ell} := c_{\ell} - 1; \ c_{\ell'} := c_{\ell'} + 1 \\
  \end{array}} g'
  \]

- Creation of a new thread at initial state \( \ell \):
  \[
  g \xrightarrow{\begin{array}{c}
c_{\ell} := c_{\ell} + 1 \\
  \end{array}} g
  \]
VASS: Reachability Problems

- **State reachability problem:**
  
  Given a state \( q \), determine if a configuration \((q, V)\) is reachable, for some \( V \in \mathbb{N}^n \) (any one).

- **Coverability problem:**
  
  Given a configuration \((q, V)\), determine if a configuration \((q, V')\) is reachable, for some \( V' \geq V \). (We say that \((q, V)\) is coverable.)

  EXSPACE-complete [Rackoff 78]

  NB: **Coverability can be reduced to State reachability and vice-versa.**

- **Configuration reachability problem:**
  
  Determine if a given configuration \((q, V)\) is reachable.

  Decidable [Mayr 81], [Kosaraju 82].

  EXPSPACE-hard [Lipton 75]. No upper bound known.
Well Structured Systems

[Abdulla et al. 96], [Finkel, Schnoebelen, 00]

- Let $U$ be a universe.
- Well-quasi ordering $\preceq$ over $U$: $\forall c_0, c_1, c_2, \ldots, \exists i < j, \ c_i \preceq c_j$
- $\Rightarrow$ Each (infinite) set has a finite minor set.

- Let $S \subseteq U$. Upward-closure $\overline{S} =$ minimal subset of $U$ s.t.
  - $S \subseteq \overline{S}$,
  - $\forall x, y. (x \in S \text{ and } x \preceq y) \Rightarrow y \in \overline{S}$.
- A set is upward closed if $\overline{S} = S$
- Upward closed sets are definable by their minor sets
  - Assume there is a function $\text{Min}$ which associates a minor to each set.
  - Assume $\text{pre}(\text{Min}(S))$ is computable for each set $S$.

- Monotonicity: $\preceq$ is a simulation relation

  $\forall c_1, c'_1, c_2. \ ((c_1 \rightarrow c'_1 \text{ and } c_1 \preceq c_2) \Rightarrow \exists c'_2. c_2 \rightarrow c'_2 \text{ and } c'_1 \preceq c'_2)$
Key lemma

Lemma

*The pre and pre* images of upward closed set are upward closed*

1. Let $S$ be an upward closed set.
2. Assume $\text{pre}(S)$ is not upward closed.
3. Let $c_1 \in \text{pre}(S)$, and let $c_2 \in U$ such that $c_1 \preceq c_2$ and $c_2 \notin \text{pre}(S)$
Key lemma

Lemma

The pre and pre* images of upward closed set are upward closed.

1. Let $S$ be an upward closed set.
2. Assume $\text{pre}(S)$ is not upward closed.
3. Let $c_1 \in \text{pre}(S)$, and let $c_2 \in U$ such that $c_1 \preceq c_2$ and $c_2 \not\in \text{pre}(S)$
4. Let $c'_1 \in S$ such that $c_1 \rightarrow c'_1$
Key lemma

Lemma

The pre and pre* images of upward closed set are upward closed

1. Let $S$ be an upward closed set.
2. Assume $\text{pre}(S)$ is not upward closed.
3. Let $c_1 \in \text{pre}(S)$, and let $c_2 \in U$ such that $c_1 \preceq c_2$ and $c_2 \not\in \text{pre}(S)$.
4. Let $c'_1 \in S$ such that $c_1 \rightarrow c'_1$.
5. Monotonicity $\Rightarrow$ there is a $c'_2$ such that $c_2 \rightarrow c'_2$ and $c'_1 \preceq c'_2$.
Key lemma

Lemma

The pre and pre* images of upward closed set are upward closed

1. Let $S$ be an upward closed set.
2. Assume $\text{pre}(S)$ is not upward closed.
3. Let $c_1 \in \text{pre}(S)$, and let $c_2 \in U$ such that $c_1 \preceq c_2$ and $c_2 \notin \text{pre}(S)$
4. Let $c'_1 \in S$ such that $c_1 \rightarrow c'_1$
5. Monotonicity $\Rightarrow$ there is a $c'_2$ such that $c_2 \rightarrow c'_2$ and $c'_1 \preceq c'_2$
6. $S$ is upward closed $\Rightarrow c'_2 \in S$
Key lemma

**Lemma**

The pre and pre* images of upward closed set are upward closed

1. Let $S$ be an upward closed set.
2. Assume $\text{pre}(S)$ is not upward closed.
3. Let $c_1 \in \text{pre}(S)$, and let $c_2 \in U$ such that $c_1 \preceq c_2$ and $c_2 \notin \text{pre}(S)$
4. Let $c'_1 \in S$ such that $c_1 \rightarrow c'_1$
5. Monotonicity $\Rightarrow$ there is a $c'_2$ such that $c_2 \rightarrow c'_2$ and $c'_1 \preceq c'_2$
6. $S$ is upward closed $\Rightarrow$ $c'_2 \in S$
7. $\Rightarrow c_2 \in \text{pre}(S)$, contradiction.
Key lemma

Lemma

The pre and pre* images of upward closed set are upward closed

1. Let $S$ be an upward closed set.
2. Assume $\text{pre}(S)$ is not upward closed.
3. Let $c_1 \in \text{pre}(S)$, and let $c_2 \in U$ such that $c_1 \preceq c_2$ and $c_2 \notin \text{pre}(S)$
4. Let $c'_1 \in S$ such that $c_1 \rightarrow c'_1$
5. Monotonicity $\Rightarrow$ there is a $c'_2$ such that $c_2 \rightarrow c'_2$ and $c'_1 \preceq c'_2$
6. $S$ is upward closed $\Rightarrow$ $c'_2 \in S$
7. $\Rightarrow$ $c_2 \in \text{pre}(S)$, contradiction.
8. For $\text{pre}^*$: the union of upward closed sets is upward closed.
Backward Reachability Analysis

Consider the increasing sequence $X_0 \subseteq X_1 \subseteq X_2 \ldots$ defined by:

- $X_0 = \text{Min}(S)$
- $X_{i+1} = X_i \cup \text{Min}(\text{pre}(X_i))$

Termination:

*There is an index $i \geq 0$ such that $X_{i+1} = X_i$*

- The set $\text{pre}^*(S)$ is upward closed $\Rightarrow$ has a finite minor
- Wait until a minor is collected
- How long shall we wait?
- Non primitive recursive in general
The case of VASS

- Usual $\leq$ order over $\mathbb{N}$ is a WQO (Dickson lemma)
- Product of WQO’s is a WQO.
- $\Rightarrow \leq$ generalized to $\mathbb{N}^n$ is a WQO.
The case of VASS

- Usual $\leq$ order over $\mathbb{N}$ is a WQO (Dickson lemma)
- Product of WQO’s is a WQO.
- $\Rightarrow \leq$ generalized to $\mathbb{N}^n$ is a WQO.
- Upward-closed sets = finite disjunctions of $\bigwedge_{i=1}^{n} l_i \leq c_i$, where $l_i \in \mathbb{N}$
- Computation of the Pre:
  - $\text{op} = "c_j := c_j + 1" : (\bigwedge_{i \neq j} l_i \leq c_i) \land (\max(l_j - 1, 0) \leq c_j)$
  - $\text{op} = "c_j > 0/c_j - 1": (\bigwedge_{i \neq j} l_i \leq c_i) \land (l_j + 1 \leq c_j)$
The case of VASS

- Usual $\leq$ order over $\mathbb{N}$ is a WQO (Dickson lemma).
- Product of WQO’s is a WQO.
- $\Rightarrow \leq$ generalized to $\mathbb{N}^n$ is a WQO.
- Upward-closed sets = finite disjunctions of $\bigwedge_{i=1}^{n} l_i \leq c_i$, where $l_i \in \mathbb{N}$
- Computation of the Pre:
  - $op = "c_j := c_j + 1" : (\bigwedge_{i \neq j} l_i \leq c_i) \land (\max(l_j - 1, 0) \leq c_j)$
  - $op = "c_j > 0/c_j - 1" : (\bigwedge_{i \neq j} l_i \leq c_i) \land (l_j + 1 \leq c_j)$
- No test to zero, only guards of the form $c > 0 \Rightarrow$ Monotonicity
- $\Rightarrow$ Coverability is decidable.
The case of VASS

- Usual $\leq$ order over $\mathbb{N}$ is a WQO (Dickson lemma)
- Product of WQO’s is a WQO.
- $\Rightarrow \leq$ generalized to $\mathbb{N}^n$ is a WQO.
- Upward-closed sets = finite disjunctions of $\bigwedge_{i=1}^n l_i \leq c_i$, where $l_i \in \mathbb{N}$
- Computation of the Pre:
  - $op = "c_j := c_j + 1" : \ (\bigwedge_{i \neq j} l_i \leq c_i) \land (\max(l_j - 1, 0) \leq c_j)$
  - $op = "c_j > 0/c_j - 1" : \ (\bigwedge_{i \neq j} l_i \leq c_i) \land (l_j + 1 \leq c_j)$

- No test to zero, only guards of the form $c > 0 \Rightarrow$ Monotonicity
- $\Rightarrow$ Coverability is decidable.
- Can we have operation of the following forms? :

$$c_i := 0, \ c_i := c_j, \ c_i := c_i + c_j, \ c_i := c_j + c_k$$
The case of VASS

- Usual $\leq$ order over $\mathbb{N}$ is a WQO (Dickson lemma)
- Product of WQO’s is a WQO.
- $\Rightarrow \leq$ generalized to $\mathbb{N}^n$ is a WQO.
- Upward-closed sets = finite disjunctions of $\bigwedge_{i=1}^n l_i \leq c_i$, where $l_i \in \mathbb{N}$
- Computation of the Pre:
  - $op = “c_j := c_j + 1” : (\bigwedge_{i \neq j} l_i \leq c_i) \land (\max(l_j - 1, 0) \leq c_j)$
  - $op = “c_j > 0/c_j - 1” : (\bigwedge_{i \neq j} l_i \leq c_i) \land (l_j + 1 \leq c_j)$
- No test to zero, only guards of the form $c > 0 \Rightarrow$ Monotonicity
- $\Rightarrow$ Coverability is decidable.
- Can we have operation of the following forms? :
  - $c_i := 0, c_i := c_j, c_i := c_i + c_j, c_i := c_j + c_k$
- Coverability is still decidable. (But not reachability. [Dufourd et al. 98])
The case of Lossy Fifo Channel Systems

- Subword relation over a finite alphabet is a WQO (Higman’s lemma)
The case of Lossy Fifo Channel Systems

- Subword relation over a finite alphabet is a WQO (Higman's lemma)
- Upward-closed sets = finite unions of
  \[ \Sigma^* a_1 \Sigma^* a_2 \cdots a_m \Sigma^* \]

- Computation of the Pre:
  - **Send**: Left concatenation + Upward closure
  - **Receive**: Right derivation

Lossyness \(\Rightarrow\) Monotonicity \(\Rightarrow\) Coverability is decidable.

Is configuration reachability decidable?
Yes, lossyness \(\Rightarrow\) (reachability \(\simeq\) coverability)
The case of Lossy Fifo Channel Systems

- Subword relation over a finite alphabet is a WQO (Higman’s lemma)
- Upward-closed sets = finite unions of
  \[ \Sigma^* a_1 \Sigma^* a_2 \cdots a_m \Sigma^* \]

- Computation of the Pre:
  - Send: Left concatenation + Upward closure
  - Receive: Right derivation

- Lossyness \(\Rightarrow\) Monotonicity

\(\Rightarrow\) Coverability is decidable.
The case of Lossy Fifo Channel Systems

- Subword relation over a finite alphabet is a WQO (Higman’s lemma)
- Upward-closed sets = finite unions of
  $$\Sigma^* a_1 \Sigma^* a_2 \cdots a_m \Sigma^*$$

- Computation of the Pre:
  - Send: Left concatenation + Upward closure
  - Receive: Right derivation

- Lossyness $\Rightarrow$ Monotonicity

- $\Rightarrow$ Coverability is decidable.

- Is configuration reachability decidable?
The case of Lossy Fifo Channel Systems

- Subword relation over a finite alphabet is a WQO (Higman’s lemma)
- Upward-closed sets = finite unions of
  \[ \Sigma^* a_1 \Sigma^* a_2 \cdots a_m \Sigma^* \]
- Computation of the Pre:
  - Send: Left concatenation + Upward closure
  - Receive: Right derivation
- Lossyness ⇒ Monotonicity
- ⇒ Coverability is decidable.
- Is configuration reachability decidable?
- Yes, lossyness ⇒ (reachability ≃ coverability)
Concurrent Programs with Procedures

- Procedural program $\rightarrow$ Pushdown System (finite control + stack)
- Concurrent program $\rightarrow$ Concurrent PDS’s (Multistack systems)
- Two stacks can simulate a Turing tape.
- Concurrent programs with 2 threads are Turing powerful.

$\Rightarrow$ Restrictions
  - Classes of programs with particular features
  - Particular kind of behaviors
    (under-approximate analysis for bug detection)
Asynchronous Programs

- **Synchronous calls**
  
  *Usual procedure calls*

- **Asynchronous calls**
  
  - Calls are stored and dispatched later by the scheduler
  - They can be executed in any order

- Event-driven programming (requests, responses)
- Useful model: distributed systems, web servers, embedded systems
A task is a sequential (pushdown) process with dynamic task creation.

Created tasks are stored in an unordered buffer (multiset).

Tasks run until completion.

If the stack is empty, a task in moved from the multiset to the stack.
Difficulties

- Unbounded buffer of tasks
- The buffer is a multiset $\Rightarrow$ can be encoded as counters
- Need to combine somehow PDS with VASS
- Stack $\Rightarrow$ not Well Structured
- How to get rid of the stack?
State Reachability of Multiset PDS

**Theorem**

The control state reachability problem for MPDS is \textit{EXPSPACE}-complete.

\textit{Reduction to/from the coverability problem for Petri.}

First decidability proof by K. Sen and M. Viswanathan, 2006
Semi-linear Sets

- Linear set over $\mathbb{N}^n$ is a set of the form

$$\{ \vec{u} + k_1 \vec{v}_1 + \cdots + k_m \vec{v}_m : k_1, \ldots, k_m \in \mathbb{N} \}$$

where $\vec{u}, \vec{v}_1, \ldots, \vec{v}_m \in \mathbb{N}^n$

- Semi-linear set = finite union of linear sets.

- Examples:
  - $\{(0,0) + k(1,1) : k \geq 0\} \equiv x_1 = x_2$
  - $\{(0,0) + k(1,2) : k \geq 0\} \equiv 2x_1 = x_2$
  - $\{(0,3) + k(1,1) : k \geq 0\} \equiv x_1 + 3 = x_2$
  - $\{(0,3) + k_1(0,1) + k_2(1,1) : k \geq 0\} \equiv x_1 + 3 \leq x_2$
  - $\{(0,0,0) + k_1(1,0,1) + k_2(0,1,1) : k_1, k_2 \geq 0\} \equiv x_1 + x_2 = x_3$
  - $\{(0,0,3) + k_1(1,0,2) + k_2(0,1,1) : k_1, k_2 \geq 0\} \equiv 2x_1 + x_2 + 3 = x_3$

Theorem [Ginsburg, Spanier, 1966]
A set is semi-linear iff it is definable in Presburger arithmetics.
Semi-linear Sets

- Linear set over $\mathbb{N}^n$ is a set of the form
  \[
  \{ \vec{u} + k_1 \vec{v}_1 + \cdots + k_m \vec{v}_m : k_1, \ldots, k_m \in \mathbb{N} \}
  \]
  where $\vec{u}, \vec{v}_1, \ldots, \vec{v}_m \in \mathbb{N}^n$

  - Semi-linear set = finite union of linear sets.

- Examples:
  - $\{(0,0) + k(1,1) : k \geq 0\} \equiv x_1 = x_2$
  - $\{(0,0) + k(1,2) : k \geq 0\} \equiv 2x_1 = x_2$
  - $\{(0,3) + k(1,1) : k \geq 0\} \equiv x_1 + 3 = x_2$
  - $\{(0,3) + k_1(0,1) + k_2(1,1) : k \geq 0\} \equiv x_1 + 3 \leq x_2$
  - $\{(0,0,0) + k_1(1,0,1) + k_2(0,1,1) : k_1, k_2 \geq 0\} \equiv x_1 + x_2 = x_3$
  - $\{(0,0,3) + k_1(1,0,2) + k_2(0,1,1) : k_1, k_2 \geq 0\} \equiv 2x_1 + x_2 + 3 = x_3$

- Theorem [Ginsburg, Spanier, 1966]
  \[\text{A set is semi-linear iff it is definable in Presburger arithmetics.}\]
Let $\Sigma = \{a_1, \ldots, a_n\}$.

Given a word $w \in \Sigma^*$, the Parikh image of $w$ is:

$$\phi(w) = (\#a_1(w), \ldots, \#a_n(w)) \in \mathbb{N}^n$$

Given a language $L \subseteq \Sigma^*$, $\phi(L) = \{\phi(w) : w \in L\}$

Examples:

- $L_1 = \{a^n b^n : n \geq 0\}$, $\phi(L_1) = \{(x_1, x_2) : x_1 = x_2\}$
- $L_2 = \{a^n b^n c^n : n \geq 0\}$, $\phi(L_2) = \{(x_1, x_2, x_3) : x_1 = x_2 \land x_2 = x_3\}$
- $L_3 = (ab)^* = \{(ab)^n : n \geq 0\}$, $\phi(L_3) = \{(x_1, x_2) : x_1 = x_2\}$
Parikh’s Theorem (1966)

*For every Context-Free Language L, $\phi(L)$ is a semi-linear set.*
Semi-linear sets, CFL’s, and RL’s

- **Parikh’s Theorem (1966)**
  
  \[ \text{For every Context-Free Language } L, \, \phi(L) \text{ is a semi-linear set.} \]

- **Proposition**
  
  \[ \text{For every semi-linear set } S, \text{ there exists a Regular Language } L \text{ such that } \phi(L) = S. \]

- **Corollary**
  
  \[ \text{For every Context-Free Language } L, \text{ there exists a Regular language } L' \text{ such that } \phi(L) = \phi(L'). \]
From Multiset PDS to VASS

PDS computation with tasks creation

$q_0 \gamma_0 \quad q_1 \gamma_1 \quad q_2$

Pending tasks Multiset

$\text{Parikh's Theorem: } M_i \text{ is definable by a finite state automaton } S_i$

Construction of a VASS:

Simulation of $S_i + \text{task consumption rules}$
From Multiset PDS to VASS

PDS computation with tasks creation

Pending tasks Multiset

Parikh's Theorem:

M_i is definable by a finite state automaton

Construction of a VASS:

Simulation of S_i + task consumption rules
From Multiset PDS to VASS

PDS computation with tasks creation

$q_0 \rightarrow \gamma_0 \rightarrow q_1 \rightarrow \gamma_1 \rightarrow q_2$

Pending tasks Multiset

$M_1$

$M_1$ is the Parikh image of $L_1$.

$L_1$ is a Context-Free Language.

$L_1 = \text{Set of sequences of created tasks}$.

Parikh's Theorem: $M_i$ is definable by a finite state automaton $S_i$.

Construction of a VASS: Simulation of $S_i$ + task consumption rules.
From Multiset PDS to VASS

PDS computation with tasks creation

Pending tasks Multiset
From Multiset PDS to VASS

PDS computation with tasks creation

$q_0 \gamma_0 \rightarrow q_1 \gamma_1 \rightarrow q_2$

Pending tasks Multiset

$M_1$ and $M_2$ are the Parikh images of $L_1$ and $L_2$, respectively.

Parikh's Theorem:
Parikh's Theorem: $M_i$ is definable by a finite state automaton $S_i$.

Construction of a VASS:
Simulation of $S_i$ with task consumption rules.
From Multiset PDS to VASS

$L_1 = \text{Set of sequences of created tasks}$

$L_1$ is a Context-Free Language

$M_1$ is the Parikh image of $L_1$
From Multiset PDS to VASS

Parikh’s Theorem: $M_i$ is definable by a finite state automaton $S_i$
From Multiset PDS to VASS

Parikh’s Theorem: \( M_i \) is definable by a finite state automaton \( S_i \)

Construction of a VASS: Simulation of \( S_i \) + task consumption rules
Message-Passing Programs with Procedures

- Undecidable even for bounded channels
- Restrictions on
  - Interaction between recursion and communication (e.g., communication with empty stack)
  - Kind of channels (e.g., lossy, unordered)
  - Topology of the network
- Decidable classes
  
  [La Torre et al. TACAS’08], [Atig et al., CONCUR’08], …
A simple case: Acyclic Lossy Channel Pushdown Networks

- Consider the system $P_1 \xrightarrow{c_1} P_2 \xrightarrow{c_2} P_3 \cdots P_{n-1} \xrightarrow{c_{n-1}} P_n$
- Problem: Is it possible to reach the global state $(q_1, q_2, \ldots, q_n)$?
A simple case: Acyclic Lossy Channel Pushdown Networks

- Consider the system $P_1 \xrightarrow{c_1} P_2 \xrightarrow{c_2} P_3 \cdots P_{n-1} \xrightarrow{c_{n-1}} P_n$
- Problem: Is it possible to reach the global state $(q_1, q_2, \ldots, q_n)$?
- Consider the set $L(c_1)$ of all possible contents of $c_1$ resulting from $P_1$ computations reaching $q_1$
A simple case: Acyclic Lossy Channel Pushdown Networks

- Consider the system $P_1 \xrightarrow{c_1} P_2 \xrightarrow{c_2} P_3 \cdots P_{n-1} \xrightarrow{c_{n-1}} P_n$

- Problem: Is it possible to reach the global state $(q_1, q_2, \ldots, q_n)$?

- Consider the set $L(c_1)$ of all possible contents of $c_1$ resulting from $P_1$ computations reaching $q_1$

- This set is downward closed w.r.t. the subword relation.

- Downward closed sets are regular: unions of

$$
\Sigma_1^* (a_1 + \epsilon) \cdots (a_m + \epsilon) \Sigma_{m+1}^*
$$
A simple case: Acyclic Lossy Channel Pushdown Networks

- Consider the system $P_1 \xrightarrow{c_1} P_2 \xrightarrow{c_2} P_3 \cdots P_{n-1} \xrightarrow{c_{n-1}} P_n$
- Problem: Is it possible to reach the global state $(q_1, q_2, \ldots, q_n)$?
- Consider the set $L(c_1)$ of all possible contents of $c_1$ resulting from $P_1$ computations reaching $q_1$
- This set is downward closed w.r.t. the subword relation.
- Downward closed sets are regular: unions of $\Sigma_1^*(a_1 + \epsilon) \cdots (a_m + \epsilon)\Sigma_{m+1}^*$
- The downward closure of a CFL is effectively constructible [Courcelle, 91]
A simple case: Acyclic Lossy Channel Pushdown Networks

- Consider the system $P_1 \xrightarrow{c_1} P_2 \xrightarrow{c_2} P_3 \cdots P_{n-1} \xrightarrow{c_{n-1}} P_n$
- Problem: Is it possible to reach the global state $(q_1, q_2, \ldots, q_n)$?
- Consider the set $L(c_1)$ of all possible contents of $c_1$ resulting from $P_1$ computations reaching $q_1$
- This set is downward closed w.r.t. the subword relation.
- Downward closed sets are regular: unions of 

$$\Sigma_1^*(a_1 + \epsilon) \cdots (a_m + \epsilon)\Sigma_{m+1}^*$$

- The downward closure of a CFL is effectively constructible [Courcelle, 91]
- Compose $L(c_1)$ with $P_2$ to get a new PDS $\tilde{P}_2$
- Solve the same problem for $\tilde{P}_2 \xrightarrow{c_2} P_3 \xrightarrow{c_3} \cdots P_{n-1} \xrightarrow{c_{n-1}} P_n$
A simple case: Acyclic Lossy Channel Pushdown Networks

- Consider the system $P_1 \xrightarrow{c_1} P_2 \xrightarrow{c_2} P_3 \cdots P_{n-1} \xrightarrow{c_{n-1}} P_n$
- Problem: Is it possible to reach the global state $(q_1, q_2, \ldots, q_n)$?
- Consider the set $L(c_1)$ of all possible contents of $c_1$ resulting from $P_1$ computations reaching $q_1$
- This set is downward closed w.r.t. the subword relation.
- Downward closed sets are regular: unions of $\Sigma_1^*(a_1 + \epsilon) \cdots (a_m + \epsilon)\Sigma_{m+1}^*$
- The downward closure of a CFL is effectively constructible [Courcelle, 91]
- Compose $L(c_1)$ with $P_2$ to get a new PDS $\tilde{P}_2$
- Solve the same problem for $\tilde{P}_2 \xrightarrow{c_2} P_3 \xrightarrow{c_3} \cdots P_{n-1} \xrightarrow{c_{n-1}} P_n$
- At the end, we need to solve reachability in one pushdown system $\tilde{P}_n$
End of Lecture 1:

- Dynamic networks of processes can be represented using VASS
- Procedures make things more difficult
- Constraints on interaction between concurrency and recursion are necessary to get decidable classes
- Asynchronous is an important class of programs for which verification problems are decidable
- Reasoning about interfaces/summaries is an important tool for the design of decision procedures
- Still, complexity is high. Need of efficient techniques.