Constraint-based Model Checking

Towards Infinite-state Model Checking

Given an *infinite* structure M a state s and a CTL property φ, does $M, s \models \varphi$ hold?

Let's try to reformulate the CTL framework here:

- **Symbolic Representation**
  State = assignment to variables of heterogenous type (*bool*, *int*, ...)
  Transition relation = **?**
  Predecessor relation *Pre* = **?**
- **Model Checking Algorithms (?)**
  Fixpoint computation using **?** as symbolic representation of *infinite* sets of states

Constraint Systems

Fixed an interpretation domain $\mathcal{D}$, a Constraint System is a tuple $\langle \mathcal{C}, \sqsubseteq \rangle$ such that

- $\mathcal{C}$ is a denumerable set of constrains
- the denotation $[\![\varphi]\!]$ of $\varphi \in \mathcal{C}$ is a subset of $\mathcal{D}$
- The entailment relation $\sqsubseteq$ is an ordering between constraints in $\mathcal{C}$ such that $\varphi \sqsubseteq \psi$ implies $[\![\psi]\!] \subseteq [\![\varphi]\!]$

Constraints as Assertional Language

We use constraints to represent infinite sets of states.

- **Minimal requirements for Reachability Properties**
  The property and the initial states are expressible in $\mathcal{C}$
  Entailment of constraints is decidable
  There is algorithm for computing $Pre_\exists$
- **Symbolic Representation**
  Transition relation $=$ Disjunction of constraints
  Predecessor relation $=$ Disjunction of *existentially quantified* constraints

Orderings on Sets of Constraints

The entailment relation $\sqsubseteq_S$ is defined as the following ordering between finite sets of constraints:

- $S \sqsubseteq S'$ iff for each $\psi \in S'$ there exists $\varphi \in S$ s.t. $\varphi \sqsubseteq \psi$
- $S \sqsubseteq S'$ implies $[\![S']\!] \subseteq [\![S]\!]$ where $[\![S]\!]$ is the natural extension of $[\![\cdot]\!]$ to sets of constraints

Examples of Assertional Languages

- Boolean Constraints
  OBDDs [Bryant]
- Presburger Arithmetics (Integer Linear Constraints)
  Omega Library
- Linear Arithmetic Constraints over Reals
  Polyhedra Libraries)
- Composite Constraints = BDD + Presburger Arithmetics
  Action Language Verifier (ALV)
- Automata
  Word and Tree Automata [Regular model checking]

Constraint-based Backward Reachability

Goal is to prove $\mathbf{AG}(\neg B) = \neg\mathbf{EF}(B)$, i.e., from states in $[\![\varphi_0]\!]$ we cannot reach states in $[\![B]\!]$, where B is a set of constraints that represents "bad states"

- We compute $Pre^*(B)$, using $\sqsubseteq_s$ to discard redundant constraints
- If the computation terminates, we check $[\![\varphi_0]\!] \cap [\![\mathbf{Pre}^*(B)]\!] = \emptyset$
- Termination is not guaranteed in general!
- Tools like HyTech and ALV may not terminate

A General Framework for Termination

- The use of the theory of *well-quasi orderings* combined with *constraints* as symbolic representation of infinite set of states leads to many interesting classes of decidable verification problems
- Some examples are
    - Lossy FIFO Channel Systems
    - Parameterized Systems
    - Timed Automata
    - Petri Nets
    - Timed Petri Nets
    - Data Nets

Well-quasi Ordering (wqo)

- A quasi (reflexive and transitive) ordering $\langle A, \preceq \rangle$ is a well-quasi ordering (wqo) if for any infinite sequence of elements $a_0 a_1 a_2 \ldots$ there exist $i < j$ s.t. $a_i \preceq a_j$
- A wqo is
  - well-founded (it does not contain infinite strictly decreasing sequences)
  - it has no infinite antichains (sequences of pairwise incomparable elements)

Examples of wqo

- For a finite set A, $\langle A, = \rangle$ is a wqo
- $\langle Nat, \leq \rangle$ is a wqo

Examples of NON wqo

- $\langle Int, \leq \rangle$ is NOT a wqo
  (it is not well founded)
- $\langle Nat, | \rangle$ where $n|m$ iff if n divides m without remainder is NOT a wqo
  (prime numbers form an antichain)
- The lexicographic order is NOT a wqo

Dickson's Lemma

- *Nat*: natural numbers
- Nat^k: tuples of k natural numbers
- $\langle a_1, \ldots, a_k \rangle \preceq \langle b_1, \ldots, b_k \rangle$ iff $a_i \leq b_i$ for $i : 1, \ldots, k$ is a wqo

Higman's Lemma: Finite Sets

- Let $\langle A, \preceq \rangle$ be a WQO
- $FSet(A)$ be the set of finite sets of elements in A
- $B = \{a_1, \ldots, a_n\} \sqsubseteq_s B' = \{a'_1, \ldots, a'_m\}$ iff there exists injective and monotonic $h : [1, \ldots, n] \to [1, \ldots, m]$ s.t. $a_i \preceq a_h(i)'$ for $i : 1, \ldots, n$
- $\langle FSet(A), \sqsubseteq_s \rangle$ is a wqo

Higman's Lemma: Bags

- Let $\langle A, \preceq \rangle$ be a wqo
- $Bag(A)$ be the set of multisets with elements in A.
- $B = [a_1, \ldots, a_n] \sqsubseteq_b B' = [a'_1, \ldots, a'_m]$ iff there exists injective $h : [1, \ldots, n] \to [1, \ldots, m]$ s.t. $a_i \preceq a_h(i)'$ for $i : 1, \ldots, n$
- $\langle Bag(A), \sqsubseteq_b \rangle$ is a wqo

Higman's Lemma: Words

- Let $\langle A, \preceq \rangle$ be a wqo
- $Word(A)$ be the set of words with elements in A.
- $B = a_1 \cdot \ldots \cdot a_n \sqsubseteq_w B' = a'_1 \cdot \ldots \cdot a'_m$ iff there exists injective and monotonic $h : [1, \ldots, n] \to [1, \ldots, m]$ s.t. $a_i \preceq a_h(i)'$ for $i : 1, \ldots, n$
- $\langle Word(A), \sqsubseteq_w \rangle$ is a wqo

Applications of Higman's Lemma

- Let Σ be a finite alphabet

- Σ^*: finite words over Σ
- $v \preceq w$ defined as v is a subword of w is a wqo

- Σ^B: finite bags over Σ
- $B \preceq B'$ defined as B is a submultiset of B' is a wqo

More on Finite Sets

- Let $\langle A, \preceq \rangle$ be a wqo
- $FSet(A)$ be the set of finite sets of elements in A.
- $B = \{a_1, \ldots, a_n\} \sqsubseteq_s B' = \{a'_1, \ldots, a'_m\}$ iff there exists $h : [1, \ldots, m] \to [1, \ldots, n]$ s.t. $a_{h(j)} \preceq a'_j$ for $j : 1, \ldots, m$
- $\langle FSet(A), \sqsubseteq_s \rangle$ is not always a wqo

Other Examples

- Kruskal's Theorem: Embedding between finite trees with nodes labeled by elements of a wqo
- Robertson-Seymour's Theorem: Finite graphs ordered by the graph minor relation is a wqo
- Ding's Theorem: Finite graphs with bounded paths ordered by the (induced) subgraph relation

Back to Constraint-based MC: Property

Let $\langle \mathcal{C}, \sqsubseteq \rangle$ be a constraint system in which $\sqsubseteq$ is a wqo

- Let $S_i \subseteq \mathcal{C}$ for $i \geq 0$
- for each infinite chain $S_0 \subseteq S_1 \subseteq S_2 \subseteq \ldots S_i \ldots$, there exists $i < j$ s.t. $S_i \sqsubseteq_s S_j$

It only works for increasing chains (not generic sequences)

Constraint-based Backward Reachability

Assumptions:

- $\langle \mathcal{C}, \sqsubseteq \rangle$ is a wqo
- $\mathcal{C}$ is closed under application of $Pre_\exists$,
- there is an algorithm to compute $Pre_\exists$ for any $S \subseteq \mathcal{C}$
  - It is often the case that $Pre(\{\varphi_1, \ldots, \varphi_n\}) = \bigcup_{t \in T, i \in [1,\ldots,n]} Pre_t(\varphi_i)$
- there is an algorithm to check $[\![\varphi]\!] \cap [\![S]\!] = \emptyset$ for any $\varphi \in \mathcal{C}$ and $S \subseteq \mathcal{C}$

Then, symbolic backward reachability is guaranteed to terminate

Perfect Channel Systems

Perfect Channel Systems

- A finite number of processes communicating via FIFO channels
- Each process is finite state
- FIFO Channels are unbounded

Definition

- C is a finite set of channel names
- M is a finite set of message names
- $Act = \{\tau\} \cup \{c.send(m), c.rec(m), c.empty \mid c \in C, m \in M\}$
- A process is defined as an automata $P = \{Q, Q_0, \delta\}$, where
  - Q is a set of control states
  - $Q_0 \subseteq Q$ is a set of initial control states
  - $\delta \subseteq (Q \times Act \times Q)$ is the transition relation

Configurations with n-processes

A system configuration with n processes is a tuple

$$\gamma = \{q_1, \ldots, q_n, h\}$$

where

- $q_i \in Q$ for $i : 1, \ldots, n$ (control state of i-th process)
- $h : C \to M^*$
- $h(c)$ is the word that encodes the current content of channel c

Operational Semantics

A transition

$$\gamma = \{q_1, \ldots, q_i, \ldots, q_n, h\} \rightarrow \{q_1, \ldots, q_i', \ldots, q_n, h'\}$$

occurs when

- $\langle q_i, \tau, q_i' \rangle \in \delta$;
- $\langle q_i, c.empty, q_i' \rangle \in \delta$ and $h'(c) = h(c) = \emptyset$ (c is empty);
- $\langle q_i, c.send(m), q_i' \rangle \in \delta$ and $h'(c) = h(c) \cdot m$ (m is enqueued in c);
- $\langle q_i, c.rec(m), q_i' \rangle \in \delta$ and $h(c) = m \cdot h'(c)$ (m is dequeued from c).

where $\cdot$ = concatenation of words

Control state reachability problem

- Let $\{q_0, \ldots, q_0, h\}$ with $h(c) = \emptyset$ for each $c \in C$.
- Can we reach a configuration in which a process is in control state q?

State-space exploration?

- FIFO channels can grow unboundedly!
- E.g. a process can repeatedly send the same set of messages like in the loop $\langle q, c!m, q \rangle$
- The state-space to explore to solve the control state reachability problem is potentially infinite

Can we solve control state reachability?

- It is possible to reduce the reachability problem for counters machines to control state reachability of communicating automata
- A counter machine is defined over K counters (integer variables) $X_1, \ldots, X_K$ and has instructions to increment, decrement, test a variable $(= 0)$, and goto jumps.

Counter system $\hookrightarrow$ channel systems

We associate channel c_X to variable X:

$$X = m \quad \textit{iff} \quad c_X = a \cdot \ldots \textit{m-times} \ldots \cdot a$$

- Instruction ℓ : *if* $X = 0$ *goto* ℓ' becomes $\langle \ell, c_X.\textit{empty}, \textit{ell}' \rangle$
- Instruction $\ell : X + +$ becomes $\langle \ell, c_X.\textit{send}(a), \textit{ell}' \rangle$
- Instruction $\ell : X - -$ becomes $\langle \ell, c_X.\textit{rec}(a), \textit{ell}' \rangle$

Back to control state reachability

- A counter machine with K counters stops in location ℓ iff the corresponding system of communicating automata with one process and K channels reaches the same location
- The halting problem of counter systems is undecidable

$$\Rightarrow$$

Control state reachability of channel systems is undecidable

Lossy Channel Systems

Perfect vs Lossy Communication

- We have considered perfect communication systems
  - the order of messages is preserved
  - messages cannot get lost
- However communication channels are often "unreliable"

Unreliable Channel Systems: Unordered Channels

- Assume that the ordering is not preserved, i.e., messages can be inserted in any position in the channel

Unreliable Channel Systems: Unordered Channels

- Assume that the ordering is not preserved, i.e., messages can be inserted in any position in the channel
- Channels can be represented as bags of symbols in M

Unreliable Channel Systems: Unordered Channels

- Assume that the ordering is not preserved, i.e., messages can be inserted in any position in the channel
- Channels can be represented as bags of symbols in M
- We can still use unordered channels to encode counters! Control state reachability is still undecidable

Unreliable Channel Systems: Lossy FIFO Channels

- Messages can get lost, the order is preserved

Operational Semantics with Message Loss

We compose $\rightarrow$ (semantics with perfect channels) with a lossy step $\rightsquigarrow$:

$$\{s, h\} \Rightarrow \{t, h'\}$$
$$\textit{iff}$$
$$\{s, h\} \rightsquigarrow \{s, h_1\} \rightarrow \{t, h_2\} \rightsquigarrow \{t, h'\}$$

s.t. $h_1(c)$ is a subword of h and $h'(c)$ is a subword of $h_2(c)$ for each $c \in C$

Control State Reachability

- Can we still encode counter machines using lossy channels?

Control State Reachability

- Can we still encode counter machines using lossy channels?
- No, the encoding of counters with channels is inaccurate (we can model lossy counters)

Control State Reachability

- Can we still encode counter machines using lossy channels?
- No, the encoding of counters with channels is inaccurate (we can model lossy counters)
- Some hope to obtain an algorithm for checking control state reachability!

Observation I

- Assume $\langle s, h_1 \rangle \Rightarrow \langle s, h_2 \rangle$ and let h'_1 s.t. $h_1(c)$ is a subword of $h'_1(c)$ for every $c \in C$
- There exists $\langle s, h'_2 \rangle$ s.t. $\langle s, h'_1 \rangle \Rightarrow \langle s, h'_2 \rangle$
- In other words $\Rightarrow$ is *monotonic* w.r.t. the following ordering $\langle s, h \rangle \preceq \langle t, h' \rangle$ iff
  - $s = t$
  - $h(c)$ is a subword of $h'(c)$ for every $c \in C$

Observation II

- Target set T: any configuration of the form $\langle s, h \rangle$ where q occurs in s for an arbitrary function h (i.e. arbitrary content of channels in C)
- T is *upward closed* w.r.t. $\preceq$, i.e., if $\langle s, h \rangle \in T$ and $\langle s, h \rangle \preceq \langle s, h' \rangle$, then $\langle s, h' \rangle \in T$
- If $\langle s, h_1 \rangle \Rightarrow \langle s', h_2 \rangle \in T$, and $\langle s, h_1 \rangle \preceq \langle s, h_1' \rangle$, then $\langle s, h_1' \rangle \Rightarrow \langle s', h_2' \rangle$ and $\langle s', h_2 \rangle \preceq \langle s', h_2' \rangle$
- In other words from the monotonicity property we have that if I is an upward closed set of configurations, then $Pre(I)$ is still upward closed

Property of subword relation

The subword relation $\preceq_s$ is a *well-quasi ordering* [Higman's Lemma]

- **No bad sequences:**
  For any infinite sequence $w_1, \ldots, w_i, \ldots$ of words, there exist $i < j$ s.t. $w_i \preceq w_j$
- **Finite basis property:**
  Any upward closed set (w.r.t. $\preceq_s$) of words has a finite set of minimal elements, i.e., upward closed sets can be represented in a finite way

Target states

- $Target_q$ = upward closed set represented by the set of minimal elements of the form $\langle s, h \rangle$ where s any contains q and $h(c) = \epsilon$ for each $c \in C$ (ϵ=empty string)
- For instance, $\langle q, q', \epsilon, \epsilon \rangle$ generates all configurations of the form $\langle q, q', w, w' \rangle$ for any $w, w' \in M^*$

Predecessor Computation

- Let S be a finite set of configurations that represent the upward closed set of configurations

$$S \uparrow = \{\langle p, h\rangle \mid \langle p, h'\rangle \in S,\ \ h(c) \preceq h'(c) \ \mathit{for\ each}\ c \in C\}$$

- We can compute a finite set S' that represents the set of one-step predecessors:

$$\mathit{pre}(S) = \{\gamma \mid \gamma \Rightarrow \gamma' \in S\}$$

Predecessor Computation: Example

- Consider the configuration $\langle q, ab \rangle$ (1 process, 1 FIFO channel)
- With the transition $\langle p, !a, q \rangle$ we compute minimal elements like: $\langle p, c = ab \rangle$ (a is enqueued but then it got lost)

$$\langle p, w_1 a w_2 b w_3 \rangle \rightarrow \langle q, w_1 a w_2 b w_3 a \rangle \rightsquigarrow \langle q, w_1 a w_2 b w_3 \rangle$$

- With the transition $\langle p, ?c, q \rangle$ we compute minimal elements like $\langle p, cab \rangle$ (c must be in the head)

$$\langle p, w_1 c w_2 a w_3 b w_4 \rangle \rightsquigarrow \langle q, c w_2 a w_3 b w_4 a \rangle \rightsquigarrow \langle q, w_2 a w_3 b w_4 \rangle$$

Backward Reachability

- We can use a symbolic backward reachability algorithm:
  - Minimal configurations to represent upward closed (infinite) sets of configurations
  - We symbolically compute predecessors (stored in *Reach*)
  - We test entailment by comparing minimal configurations
- Correctness

  $\gamma_0 = \langle q_0, \ldots, q_0, \epsilon, \ldots, \epsilon \rangle \in \mathit{Reach}$ iff $\gamma_0 \Rightarrow^* \gamma \in \mathit{Target}_q$
- Termination ensured by the wqo of $\preceq$

Complexity and Other Properties

- Terrible!
- Complexity of reachability in Lossy FIFO Channel Systems is non-primitive recursive
- The approach does not work for all temporal properties, e.g., repeated reachability of a control state (i.e. visiting a state infinitely often) is undecidable

Backward vs Forward

- It is possible to use a special class of regular expressions called S.R.E. to effectively compute one-step successors $Post(S) = \{\gamma' \mid \gamma \Rightarrow \gamma', \gamma \in S\}$
- However, the are no guarantees of termination
- Forward analsis is implemented in the tool TREX developed at Liafa