Verification of temporal logics on infinite-state systems

Day 3, Lecture 1
Model-checking temporal logics on pushdown systems

Stéphane Demri and Valentin Goranko

ESSLLI 2007, Dublin, July 2007
Büchi automata on infinite words

The set of all infinite words on an alphabet $\Sigma$ is denoted by $\Sigma^\omega$.

Büchi automaton (BA) $\langle \Sigma, S, s_0, \delta, F \rangle$, is defined just like the standard non-deterministic finite automata, but it runs on infinite words. The differences:

- Run of a BA is an infinite sequence of states.
- An infinite word is accepted by a Büchi automaton if some run on that word visits an accepting state infinitely many times.
- Thus, with every Büchi automaton $A$ on an alphabet $\Sigma$ we associate its language $L_\omega(A) \subseteq \Sigma^\omega$ consisting of the infinite words accepted by the automaton.

Such languages are called $\omega$-regular.

Checking (the language of) a Büchi automaton for emptiness is easy: $L_\omega(A) \neq \emptyset$ iff $A$ has an accepting state which is reachable from an initial state and from itself.
Büchi automaton: example

Example: \( A = \langle \{a, b\}, \{s_0, s_1\}, \delta, \{s_0\}, \{s_1\} \rangle \), where
\[
\delta(s_0, a) = \{s_0\}, \quad \delta(s_0, b) = \{s_1\}, \quad \delta(s_1, a) = \{s_0\}, \quad \delta(s_1, b) = \{s_1\}
\]

accepts those infinite words in which \( a \) occurs infinitely often.
Boolean closure of $\omega$-regular languages

**Theorem:** The family of $\omega$–regular languages are closed under all Boolean operations.

Closure under unions is easy, just like for regular languages and finite automata.

For the closure under intersection, generalized Büchi automata are introduced. They have a set of sets of accepting states, and the accepting condition is that a state from each of these sets must occur infinitely often in the run.

Furthermore, complementation cannot be obtained anymore only by determinization, because *not every* Büchi automaton can be determinized.

Instead, more complex automata can be used, e.g., Müller automata, which demand as an accepting condition the set of infinitely visited states to belong to a set of designated accepting sets of states.
Büchi automata and transition systems

With every finite LTS $\mathcal{T} = \langle S, R, L \rangle$, such that $L : S \to 2^{PROP}$, and a state $s$ we associate a Büchi automaton $A_{\mathcal{T},s}$ on the alphabet $2^{PROP}$, such that $L_\omega(A_{\mathcal{T},s})$ consists precisely of the computations in $\mathcal{T}$ starting from $s$:

$$A_{\mathcal{T},s} = (2^{PROP}, S, \{s\}, \delta, S)$$

where

$$\delta(u, q) = \{v \mid uRv \text{ and } L(u) = q\}.$$
Automata-based approach to model-checking and satisfiability checking

The idea in a nutshell: consider automata which operate on models being finite or infinite words (or trees) in some formal language.

With every formula of the logic, associate an automaton which accepts exactly those computations (words, trees) on which the formula is true.

This translates the satisfiability problem for a formula $\varphi$ to the nonemptiness problem for the language $L(A_\varphi)$.

Furthermore, the model-checking problem for $\varphi$ can now be rephrased as follows:

$T, s \models \varphi$ iff $L_\omega(A_T,s) \subseteq L(A_\varphi)$ iff $L_\omega(A_T,s) \cap L(A_{\neg \varphi}) = \emptyset$.

Now, standard automata-theoretic algorithms solving nonemptiness and containment problems for $\omega$-regular languages are applied to provide the desired satisfiability checking procedure.
Linear LTL-models are infinite words over $2^\text{PROP}$, hence Büchi automata can be considered as running over LTL-models.

**Theorem**: For every LTL-formula $\varphi$ there is a Büchi automaton $A_\varphi$ on the set of states $2^\text{PROP}$ accepting precisely the models of $\varphi$.

*Sketch of proof:*

Restrict PROP to those atomic propositions occurring in $\varphi$.

A *structural approach*: build the automaton inductively on the structure of $\varphi$. Not good, too high complexity.

This can still be done efficiently, but with alternating automata.

Instead, we build the automaton from components of the formula, taking care of specific tasks of the model-checking.
The Büchi automaton as a model checker

The idea of the automaton $A_\varphi$:

- the states are those subsets of $\text{cl}(\varphi)$, that can be true at the respective states of an LTL model.

- While running on an LTL model, $A_\varphi$ does model checking for $\varphi$ in it.
The construction of the automaton $\mathcal{A}_\varphi$

Three requirements must be checked:

1. The set of formulae regarded as true at any state must be propositionally consistent.
2. Every state must satisfy what is prescribed to it by its predecessor, e.g., every $\psi$ such that $X\psi$ is true at the predecessor.
3. All eventualities regarded as true by any state must be eventually satisfied. That is, all promises made throughout the model must be eventually fulfilled.

The automaton $\mathcal{A}_\varphi$ will be composed out of two Büchi automata taking care respectively of the 2nd and 3rd requirement, while the first one will be satisfied by the construction. These two component automata are:

- the local automaton $\mathcal{A}_\varphi^l$, and
- the eventuality automaton $\mathcal{A}_\varphi^e$. 
Subformula closure; recursive definition of $U$

Closure of an LTL-formula $\varphi$: $cl(\varphi)$, obtained by taking all subformulae of $\varphi$ and their negations, and then deleting the double negations, if any, occurring in the obtained formulae.

*Until* satisfies the following important equivalence, which defines it recursively:

$$
\chi U \psi \equiv \psi \lor (\psi \land X(\chi U \psi)).
$$

That equivalence allows the automaton to check the satisfaction of eventualities in the model by checking the local steps plus maintaining records of those eventualities whose satisfaction has been deferred to future states.
The local automaton: states

\[ A^l_\varphi = \langle 2^{cl(\varphi)}, N_l, \delta_l, N_\varphi, N_l \rangle \]

where:

- \( N_l \) consists of all maximal propositionally consistent subsets of \( cl(\varphi) \), i.e., the propositionally consistent subsets \( s \) satisfying the additional conditions:

For every \( \psi \in cl(\varphi) \), either \( \psi \in s \) or \( \sim \psi \in s \).

Note that every set \( s \) from \( N_l \) also satisfies the following:

- \( \bot \notin s \);
- for every \( \psi \rightarrow \chi \in cl(\varphi) \):
  \( \psi \rightarrow \chi \in s \) iff either \( \neg \psi \in s \) or \( \chi \in s \).
- for every \( \chi U \psi \in cl(\varphi) \), if \( \chi U \psi \in s \) then \( \psi \in s \) or \( \chi \in s \).

- \( N_\varphi \) consist of those sets \( s \in N_l \) which contain \( \varphi \).
- Every state is accepting.
Büchi automata on infinite words and model checking of LTL

Model Checking LTL and CTL* on pushdown systems

The local automaton: transitions

\[ \delta_l \text{ is defined as follows:} \]

\[ t \in \delta_l(s, a) \iff s = a \text{ and } t \text{ satisfies the local conditions:} \]

- for every \( X \psi \in cl(\varphi) \), if \( X \psi \in s \) then \( \psi \in t \);
- for every \( \chi U \psi \in cl(\varphi) \), if \( \chi U \psi \in s \) and \( \psi \notin s \) then \( \chi U \psi \in t \).
The eventuality automaton: definition

\[ A^e_\varphi = \langle 2^{\text{cl}(\varphi)}, 2^{\text{ev}(\varphi)}, \delta_e, \{ \emptyset \}, \{ \emptyset \} \rangle, \]

where:

- \( \text{ev}(\varphi) \) is the subsets of eventualities in \( \text{cl}(\varphi) \);
- \( \delta_e \) is defined as follows: \( t \in \delta_l(s, a) \) iff:
  
  (i) \( s = \emptyset \) and \( t \) consists of those formulae \( \chi U \psi \in A \) such that \( \psi \notin a \), or

  (ii) \( s \neq \emptyset \) and \( t \) consists of those formulae \( \chi U \psi \in s \) such that \( \psi \notin a \).
The eventuality automaton: intuition

At the beginning there are no eventualities to be satisfied.

As the execution goes, they pile up in the current state (consisting of the 'imminent tasks'), and meanwhile some of them get satisfied.

When the current state is emptied, then the automaton looks at the model to see what eventualities are still to be satisfied.

Note that it is not necessary to check the eventualities at every step, but only periodically, because all unsatisfied yet eventualities will be carried forward by the local automaton.

The model is accepted if the empty set of eventualities is visited infinitely often, i.e. they all get eventually satisfied.
The final construction

Once constructed, the two automata $A^l_\varphi$ and $A^e_\varphi$ are combined into one, viz., their product $A^l_\varphi \times A^e_\varphi$, accepting the intersection of their languages, i.e., those models which satisfy all conditions necessary for the satisfaction of $\varphi$ at their initial state.

Finally, the obtained automaton, which is defined on the alphabet $2^{cl(\varphi)}$ is transformed to one defined on $2^{PROP}$.

Besides the alphabet, only the transition relation $\delta$ changes into $\delta'$ defined as follows:

$t \in \delta'(s, a)$ iff there is $b \in 2^{cl(\varphi)}$ such that $a = b \cap PROP$ and $t \in \delta(s, b)$. 
Model checking LTL with automata: wrap up

To summarize:

Local model checking for LTL on finite state systems can be solved by reducing to checking containment of \( \omega \)-regular languages: 
\[
L_\omega(A_{\mathcal{T},s}) \subseteq L(A_\varphi).
\]
This problem, in turn, is reduced to checking emptiness of 
\[
L_\omega(A_{\mathcal{T},s}) \cap L(A_{\neg \varphi}),
\]
equivalently, checking for emptiness of the product of the two Büchi automata.

All this can be done in time exponential in the size of the formula, but polynomial in the size of the model.

Global model checking for LTL on finite state systems is reduced to local model checking.

Likewise, satisfiability of an LTL formula \( \varphi \) is equivalent to non-emptiness of its Büchi automaton \( A_\varphi \), which is easy.

Checking satisfiability and validity of an LTL formula is PSPACE-complete.
Automata-based model checking of CTL*

Two approaches:

1. Using non-deterministic Büchi automata on infinite trees; better, using alternating tree automata.

2. Reducing to model checking of LTL-formulae, by induction on the nesting depth of universal path quantifiers in CTL*-formulae, assuming that $\exists$ is definable as $\neg \forall \neg$.

   ▷ Formulae $\forall \psi$ where $\psi$ contains no path quantifiers are essentially LTL-formulae. The global model checking problem is solved by reduction to local model checking.

   ▷ The global model checking problem for formulae of path quantifier depth $n + 1$ is reduced to formulae of depth $n$, by solving the global model checking problem for the innermost subformulae of depth 1 and then replacing them with new atomic propositions true at the same states.
Pushdown systems with initial configuration

This exposition follows

Some of the proofs can be found in the earlier paper in the reader, others in this paper, and in:

We consider pushdown systems $\mathcal{P} = (P, \Gamma, \Delta, q_0, \omega)$ extended with a bottom stack symbol $\omega$, which cannot be removed from the stack, and an initial control location $q_0$.

With $\mathcal{P}$ we associate a transition graph $\mathcal{T}_\mathcal{P}$ on the configurations of $\mathcal{P}$ as before. It has an initial configuration $(q_0, \omega)$. 
Pushdown systems with regular valuations

Interpreted pushdown system (IPDS) with regular valuations of the atomic propositions:

\[ M = (P, \Gamma, \Delta, q_0, \omega, V), \]

where \((P, \Gamma, \Delta, q_0, \omega)\) is a PDS and \(V : \text{PROP} \rightarrow 2^{P \times \Gamma^*}\) evaluates every atomic proposition to a regular set of configurations.

We will outline model-checking procedure for LTL on such models, and then extend it to CTL*.
Pushdown systems with simple valuations

First, we consider a particular case of interpreted pushdown systems where the truth of atomic propositions only depends on the control state and the top symbol of the stack content.

In this case, the valuation can be defined first as a mapping $V_0 : \text{PROP} \rightarrow 2^{P \times \Gamma}$, and then extended to $V : \text{PROP} \rightarrow 2^{P \times \Gamma^*}$ by adding all configurations with the same control state and the top stack symbol.

Such valuations will be called simple. Thus, a valuation is simple, if it is a union of simple sets of configurations of the type: $\{(p, \alpha w) \mid w \in \Gamma^*\}$ for some $p \in P, \alpha \in \Gamma$.

The model-checking problems for LTL discussed here for the general case of IPDS with regular valuations model are reducible (non-trivially) to the case of IPDS with simple valuations.
Model checking problems for LTL on pushdown systems

For an IPDS $\mathcal{M} = (P, \Gamma, \Delta, q_0, \omega, V)$ and an LTL-formula $\varphi$ we consider the following model checking problems:

(I) Model checking for the initial configuration: determine whether $\mathcal{M}, (q_0, \omega) \models \varphi$.

(II) Global model checking: compute (a finitary description of) the set of configurations, reachable or not from $(q_0, \omega)$, that violate $\varphi$.

(III) Global model checking for reachable valuations: compute (a finitary description of) the set of all configurations, reachable from $(q_0, \omega)$, that violate $\varphi$.

Recall, that we represent regular sets of configurations by so called $P$-automata, which simulate the runs in the PDS $P$ by reading the stack content bottom-up.
Model checking for LTL on PDS with simple valuations: sketch of the solution

Sketch of the automata-theoretic approach to solve the local model checking problem (I) for the initial configuration, which will yield a solution of the others, too:

The problem (I) is solved in three steps as follows:

**Step 1.** The problem is reduced to the emptiness problem for so-called Büchi pushdown systems (BPDS). These are pushdown systems with a subset of accepting control locations, obtained as products of PDS and Büchi automata.

Run of a BPDS is accepting if it visits some accepting control location infinitely often.

BPDS is empty if it has no accepting run.

**Step 2.** The emptiness problem for BPDS is reduced to computing the set $\text{pre}^*(C)$ for certain regular sets of configurations $C$.

**Step 3.** The problem of computing the set $\text{pre}^*(C)$ for a regular set of configurations $C$ was discussed and solved in lecture 3.
Büchi pushdown systems

Let $\mathcal{P} = (P, \Gamma, \Delta, q_0, \omega, V)$ be an interpreted pushdown system and $\mathcal{B} = \langle 2^{PROP}, Q, q_0, \delta, F \rangle$ be a Büchi $\omega$-word automaton.

The Büchi pushdown system (BPSD) for $\mathcal{P}$ and $\mathcal{B}$ is the product

$$\mathcal{PB} = \mathcal{P} \times \mathcal{B} = (P \times Q, \Gamma, \Delta', G)$$

where

- $\rhd ((p, q), \gamma) \hookrightarrow ((p', q'), w) \in \Delta'$ iff $(p, \gamma) \hookrightarrow (p', w) \in \Delta$ and $q' \in \delta(q, \sigma)$ for $\sigma = V^{-1}((q, \gamma))$;
- $\rhd (p, q) \in G$ iff $q \in F$.

Acceptance condition for a run in a BPDS: configurations with control states in $G$ are visited infinitely often.
The Büchi pushdown system $\mathcal{PB}_{\neg \varphi}$

Now, take the Büchi pushdown system $\mathcal{BP}_{\neg \varphi} = \mathcal{P} \times \mathcal{B}_{\neg \varphi}$.

Because the valuations are simple, it can be shown that $\mathcal{P}$ has a run violating $\varphi$ if and only if $\mathcal{PB}_{\neg \varphi}$ is nonempty.

Thus:

$\triangleright$ the local model checking problem for a formula $\varphi$ is reduced to checking non-emptiness of $\mathcal{BP}_{\neg \varphi}$, and
$\triangleright$ the global model checking problem for $\varphi$ is reduced to the problem of computing the set of configurations $s$ in $\mathcal{BP}_{\neg \varphi}$ such that $\mathcal{BP}_{\neg \varphi}$ has an accepting run starting from $s$. 
Non-emptiness of $\mathcal{BP}_{\neg \varphi}$ as reachability problem

Let $\mathcal{M} = (P, \Gamma, \Delta, q_0, \omega, V)$ be an IPDS.

The head of a transition rule $(p, \gamma) \rightarrow (p', w) \in \Delta$ is the configuration $(p, \gamma)$.

A head $(p, \gamma)$ is repeating if there exists $\nu \in \Gamma^*$ such that $(p, \gamma)$ can be reached from $(p, \gamma \nu)$ by means of a sequence of transitions that visits some control location in $G$.

Let $Rep$ be the set of repeating heads, and

$$Rep \Gamma^* = \{(p, \gamma w) \mid (p, \gamma) \in Rep, w \in \Gamma^*\}.$$

**Lemma:** $\mathcal{BP}_{\neg \varphi}$ is non-empty iff $(p_0, \omega) \in pre^*(Rep \Gamma^*)$.

Therefore, (non)-emptiness of $\mathcal{BP}_{\neg \varphi}$ can be decided by computing $Rep$ and then $pre^*(Rep \Gamma^*)$. 
Computing the set of repeating heads

In order to compute \( \text{Rep} \) we construct a head reachability graph whose nodes are the heads of \( \Delta \) and edges are defined as follows:

there is an edge from \( (p, \gamma) \) to \( (p', \beta) \) if there is a rule \( (p, \gamma) \rightarrow (p'', \nu_1 \beta \nu_2) \) in \( \Delta \) such that \( (p'', \nu_1) \rightarrow^* (p', \varepsilon) \).

If either \( p \in G \) or \( (p', \varepsilon) \) can be reached from \( (p'', \nu_1) \) visiting a final control location along the way, then the edge is marked.

**Lemma:** A head is repeating if and only if it belongs to a strongly connected component of the head reachability graph that contains at least one marked edge.

Thus, computing \( \text{Rep} \) is reduced to computing \( \text{pre}^*(\{(p, \varepsilon) \mid p \in P\}) \), then finding the strongly connected components in the head reachability graph.

Computing \( \text{pre}^* \) for regular sets was discussed in lecture 3.

Note that, since \( \text{Rep} \Gamma^* \) is regular, \( \text{pre}^*(\text{Rep} \Gamma^*) \) is regular, too.
The case of regular valuations

Thus, the model checking problem for the initial configuration in a BSP with simple valuations is solved.

The general case, with regular valuations, is obtained by reduction to the case of simple valuations.

The idea in a nutshell: encode the $\mathcal{P}$-automata computing the valuations into the structure of $\mathcal{P}$, by means of a sort of product construction, and simulate them ‘on the fly’ during the computation of $\mathcal{P}$.

This technique enables evaluation of the atomic propositions occurring in the input formula by storing the vectors of their respective automata’s states in the stack of $\mathcal{P}$ and updating them after each transition.
Harvest

Thus, model checking problem (I) is solved in the general case.

Problem (II) (global model checking) has the same solution, because the automaton recognizing $\text{pre}^*(\text{Rep}\Gamma^*)$ is a finite representation of all the configurations violating $\varphi$.

Problem (III) is solved by showing that the set of reachable configurations from a regular set of configurations in $\mathcal{P}$ is regular, by a modification of the argument for $\text{pre}^*$.

The overall complexity is DEXPTIME complete in the size of the formula, and polynomial in the size of the PDS. Precise complexity bounds can be found in the references.

The global model-checking problem for $\text{CTL}^*$ is now solved by reduction to $\text{LTL}$, just like in the finite case, because every $\text{LTL}$ formula preserves regularity of the set of configurations where it is true in every IPDS with regular valuations.