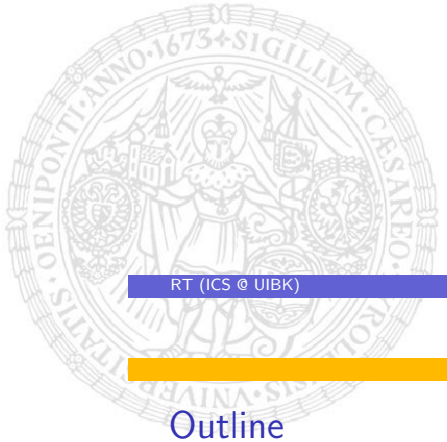
Introduction to Model Checking

René Thiemann

Institute of Computer Science
University of Innsbruck

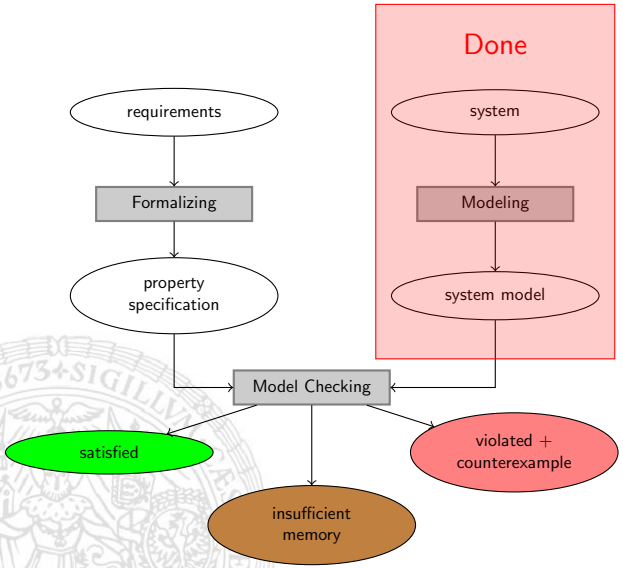
WS 2007/2008



Outline

- Regular Languages
- ω-regular Languages

Overview



Transition Systems - Revisited

A *transition system* TS is a tuple $(S, Act, \rightarrow, I, AP, L)$ where

- S is a set of states
- Act is a set of actions
- $\longrightarrow \subseteq S \times S$ is a transition relation
- $I \subseteq S$ is a set of initial states
- AP is a set of atomic propositions
- $L : S \rightarrow 2^{AP}$ is a labeling function

An execution ϱ of TS is an alternating sequence of states and actions

$$\varrho = s_0\, \alpha_1\, s_1\, \alpha_2\, \ldots\, \alpha_n\, s_n\, \ldots$$

- $s_i \xrightarrow{\alpha_{i+1}} s_{i+1}$ for all $0 \leqslant i \in \mathbb{N}$
- $s_0 \in I$

A trace of an execution is an infinite sequence of atomic propositions, i.e., $trace(\varrho) \in (2^{AP})^\omega$

$$trace(\varrho) = L(s_0)\ L(s_1)\ L(s_2)\ L(s_3)\ \ldots$$

Trace Specifications

The set of traces of a transition system $Traces(TS) \subseteq (2^{AP})^\omega$ defines the observable behaviour of TS.

- Specification: Statement about "what are the allowed behaviours"

$\Rightarrow$ A specification is an ω-language $\mathcal{L}$
($\mathcal{L}$ consists of infinite words over the alphabet 2^{AP}, $\mathcal{L} \subseteq (2^{AP})^\omega$)

- A transition system satisfies a specification $\mathcal{L}$:

$$TS \models \mathcal{L} \quad \text{iff} \quad Traces(TS) \subseteq \mathcal{L}$$

TS satisfies $\mathcal{L}$ iff all observable behaviours are allowed

- Model Checking amounts to compare ω-languages.

Question: How do we specify languages?

First consider languages $\mathcal{L}$ over finite words, $\mathcal{L} \subseteq \Sigma^*$

- Possible classes: finite, regular, context-free, context-sensitive, ...
- Model Checking requires checking $\mathcal{L}_{\text{system}} \subseteq \mathcal{L}_{\text{specification}}$
  Equivalently: $\mathcal{L}_{\text{system}} \cap \overline{\mathcal{L}_{\text{specification}}} = \varnothing$

$\Rightarrow$ Requirements on class of language

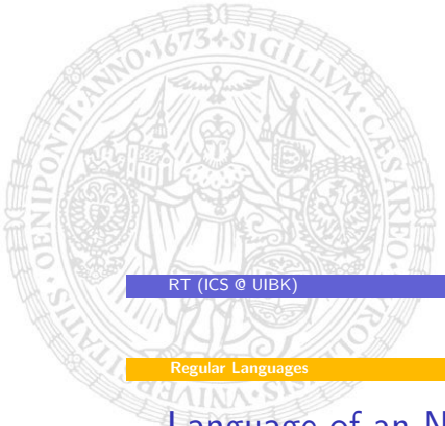
  - Closure under complement
  - Closure under intersection
  - Emptyness decidable

- Use regular languages, they are closed under all boolean operations
- Possible representations of regular languages
  - regular expressions
  - non-recursive grammars
  - finite automata

Finite Automata

A *nondeterministic finite automaton* (NFA) $\mathcal{A}$ is a tuple $(Q, \Sigma, \delta, q_0, F)$ where:

- Q is a finite set of states
- Σ is an alphabet
- $\delta : Q \times \Sigma \to 2^Q$ is a transition function
- $q_0 \in Q$ is the initial states
- $F \subseteq Q$ is a set of final (or: accepting) states

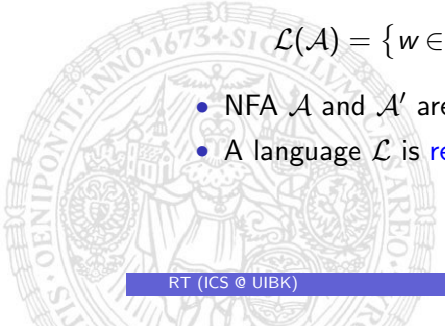


Language of an NFA

- NFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ and word $w = A_1 \dots A_n \in \Sigma^*$
- A *run* for w in $\mathcal{A}$ is a finite sequence $q_0\, q_1\, \dots\, q_n$ such that:
  - $q_i \xrightarrow{A_{i+1}} q_{i+1}$ for all $0 \leqslant i < n$
- Run $q_0\, q_1\, \dots\, q_n$ is *accepting* if $q_n \in F$
- $w \in \Sigma^*$ is *accepted* by $\mathcal{A}$ if there exists an accepting run for w
- The *accepted language* of $\mathcal{A}$:

$$\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^* \mid \text{ there exists an accepting run for } w \text{ in } \mathcal{A} \}$$

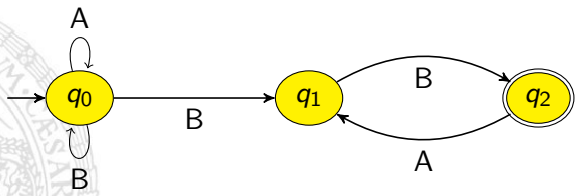
- NFA $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent* if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$
- A language $\mathcal{L}$ is regular iff $\mathcal{L} = \mathcal{L}(\mathcal{A})$ for some NFA $\mathcal{A}$



Büchi Automata

A *nondeterministic Büchi automaton* (NBA) $\mathcal{A}$ is a tuple $(Q, \Sigma, \delta, q_0, F)$ where:

- Q is a finite set of states
- Σ is an alphabet
- $\delta : Q \times \Sigma \to 2^Q$ is a transition function
- $q_0 \in Q$ is the initial states
- $F \subseteq Q$ is a set of final (or: accepting) states



Language of a NBA

- NBA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ and word $w = A_1 \dots A_n \dots \in \Sigma^\omega$
- A *run* for w in $\mathcal{A}$ is an infinite sequence $q_0\, q_1 \dots q_n \dots$ such that:
  - $q_i \xrightarrow{A_{i+1}} q_{i+1}$ for all $i \in \mathbb{N}$
- Run $q_0\, q_1 \dots q_n \dots$ is accepting if
- $w \in \Sigma^\omega$ is *accepted* by $\mathcal{A}$ if there exists an accepting run for w
- The *accepted language* of $\mathcal{A}$:

$$\mathcal{L}(\mathcal{A}) = \{ w \in \Sigma^\omega \mid \text{ there exists an accepting run for } w \text{ in } \mathcal{A} \}$$

- NBA $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent* if $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$
- A language $\mathcal{L}$ is ω-regular iff $\mathcal{L} = \mathcal{L}(\mathcal{A})$ for some NBA $\mathcal{A}$

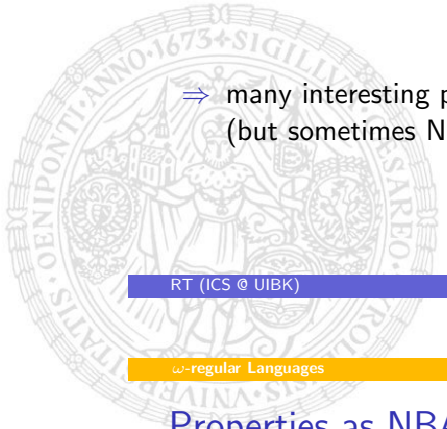
Are NBA's expressive enough?

Safety properties: (refutation by a finite prefix of an ω-word)

- always at most one traffic light is showing green
- between green and red there always is an orange phase

Liveness properties: (refutation only by whole ω-word)

- we will see green infinitely often
- whenever we select sprite then later on we will get a sprite



$\Rightarrow$ many interesting properties can be expressed by NBAs
(but sometimes NBAs are hard to understand $\Rightarrow$ need for logic)

Properties as NBAs

Model Checking with NBAs

Scenario: transition system TS and specification (as NBA $\mathcal{A}$) given

Model checking can be done in three steps

$$TS \models \mathcal{A} \text{ iff } \mathit{Traces}(TS) \subseteq \mathcal{L}(\mathcal{A}) \text{ iff } \mathit{Traces}(TS) \cap \overline{\mathcal{L}(\mathcal{A})} = \varnothing$$

- calculate NBA $\overline{\mathcal{A}}$ with $\mathcal{L}(\overline{\mathcal{A}}) = \overline{\mathcal{L}(\mathcal{A})}$
- calculate NBA $\mathcal{A}'$ for intersection of $\overline{\mathcal{A}}$ and TS
- perform non-emptyness test for $\mathcal{L}(\mathcal{A}')$

Cycles and Strongly Connected Components

Let G be a graph (V, E) with nodes V and edges E.

- A set $C \subseteq V$ is a cycle of G iff
  for all $v_1, v_2 \in C$ there is a non-empty path from v_1 to v_2
- A strongly connected component (SCC) is a maximal cycle
  (C is SCC iff $C' \supset C$ implies C' is not a cycle)
- Remarks:
  - Two SCCs C_1 and C_2 are either disjoint or identical
  - The set of SCCs of a graph can be determined in linear time (Tarjan)

Non-emptyness of $\mathcal{L}(\mathcal{A})$ for some NBA $\mathcal{A}$

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ accepts at least one ω-word

- iff there is ω-word w with accepting run $q_0\, q_1\, q_2\, \ldots$ of $\mathcal{A}$
- iff there is accepting run $q_0\, q_1\, q_2\, \ldots$ of $\mathcal{A}$
- iff there is run $q_0\, q_1\, q_2\, \ldots$ where some $q_f \in F$ occurs infinitely often
- iff in the graphical representation of $\mathcal{A}$ there is a path from q_0 to $q_f \in \mathrm{F}$ and q_f is element of an SCC
- iff in the graphical representation of $\mathcal{A}$ there is a path from q_0 to an SCC containing a final state

Hence, compute SCCs of $\mathcal{A}$ (linear time, Tarjan's algorithm)
and perform reachability-analysis from q_0 (linear time, depth first search)

- If no SCC with final state reachable from q_0: $\mathcal{L}(\mathcal{A}) = \varnothing$
- Otherwise, obtain path from q_0 to q_f and non-empty path from q_f to q_f with corresponding words w_{0f} and w_{ff}
  $\Rightarrow w_{0f} w_{ff}^{\omega} \in \mathcal{L}(\mathcal{A})$

Summary

- ω-languages specify allowed behaviour
- NFA over finite words $\rightarrow$ NBA over infinite words
  (regular languages $\rightarrow$ regular ω-languages)
- Several specifications can be formalized with NBAs
- Model Checking is decidable for specifications given as NBA

$$TS \models \mathcal{L}(\mathcal{A}) \qquad \text{iff} \qquad \mathit{Traces}(TS) \cap \overline{\mathcal{L}(\mathcal{A})} = \varnothing$$

- Complementation
- Intersection
- Non-emptyness test

Exercises

Let $\Sigma = \{A, B\}$. Construct Büchi automata for the languages:

- $\{w \mid w$ contains only finitely many A's$\}$
- $\{w \mid w$ contains only finitely many A's or infinitely many B's$\}$
- $\{w \mid w$ starts with an even number of A's followed by a $B\}$