Recap

• Infinite behaviour of DTMCs?
• Does the probability of rain on July 6, 2018 depend on today’s weather?
• Calculation of probabilities with matrices?
• Is it guaranteed that a fixed point is reached? Uniqueness of solutions?
• Finite vs. infinite DTMCs?
• What is a Markov Decision Process?
• How is MDP model checking done?
Model Checking of Software

• Typical workflows in software development

Capture Requirements  Formulate Properties  Build Model  Verify  Implement  Test

Implement and Test  Abstract Software  Formulate Properties  Verify

• What’s wrong with that?
• In order to avoid errors in the formal specification, use a high-level specification language
  - research, e.g., on controlled natural languages which can be directly translated into (temporal) logics

• In order to avoid errors by manual implementation / abstraction of the program
  - either use code generation to generate code automatically from the (verified) model, or
  - use the software as a model to be checked

• What is the difference between a model and a program? (e.g., a Java program)
Some Philosophical Questions

• Model = Kripke structure, labelled transition system, Büchi automaton, Markov chain, ...
  ▪ essence: *States* and *Transitions*

• Program = imperative, object-oriented, functional, data-flow, ...
  ▪ essence: data and control structures
  ▪ executable semantics

• What is the “state” of a program? What are program transitions?
Syntax of while-Programs

- Given a (typed) signature $\Sigma = (D, F, R)$ (domains, functions, relations), and a (denumerable) set $\mathcal{V}$ of program variables.
  - (each program variable has a type)
  - ($\mathcal{T}$ is the set of terms in the signature)
  - for simplicity, assume always $R$ contains equality $\equiv$

- A while-program is defined as follows

  $\text{whileProg} ::= \text{skip} \mid \mathcal{V}=\mathcal{T} \mid \{\text{whileProg}; \text{whileProg}\} \mid$
  
  $\text{if (FOL\textsuperscript{-}) whileProg else whileProg} \mid$
  
  $\text{while (FOL\textsuperscript{-}) whileProg}$

  where FOL\textsuperscript{-} is a quantifier-free first-order formula over $(\Sigma, \mathcal{V})$
Examples

- $\Sigma = (\{\text{int}\}, \{0, \%, \text{==}\}, \{\text{==}\}), \mathcal{V} = (a, b, c)$
  - $\Pi_1 = \text{while} (\neg a==0) \{\{c = a; a = b\%a\}; \ b = c\}$
  - $\Pi_2 = \text{if} (0==((a\%0)\%a) \text{ skip else \{skip;skip\}}$

- $\Sigma = (\{\text{int}\}, \{0, 1, 48, +, - , **\}, \{<, \text{isprim}\}), \mathcal{V} = (n, k)$
  - $\Pi_3 = \text{if} (\text{isprim}(n)) \ n=k$
  - $\Pi_{\text{Mersenne}} = \{n=0; k=0; \text{while} (k<48) \{n++; \text{if} (\text{isprim}((2**n)-1)) \ k++\}\}$
Semantics

- What is the “meaning” of such a program?
  - e.g., $\Pi_3 = \text{if } (\text{isprim}(n)) \text{ k=n}$
- need a first-order model $M: (U, I, V)$ for $(\Sigma, V)$
  - e.g., $U=\{\text{zero, one, two, three, ...}\}$, $I(0)=\text{zero, } I(1)=\text{one, ...}$, $I(\text{isprim})=\{\text{two, three, five, ...}\}$, $V(n)=\text{two, } V(k)=\text{zero}$
- Program modifies states (valuations)
  - $V'(n)=\text{two, } V'(k)=\text{two}$
- Three basic types of semantics (denotational, logical, operational)
  - structured operational semantics constructs a Kripke model to a while program
Structured Operational Semantics

- Simulates the operations of a “real” machine
  - transitions from valuation to valuation
  - program counter is increased with the program
- Abstract representation:
  - state=(program, valuation)
    - program means the part which is still to be executed
  - transition=(state1, state2)
- “Meaning” of a program is a (possibly infinite) set of such transitions (i.e., a Kripke structure)
- Execution of a program = path through the model
SOS-Rules

- \((v=t, V) \rightarrow (\text{skip, } V[v:=t])\);

- \((\{\text{skip; } \Pi\}, V) \rightarrow (\Pi, V)\)

- if \((\Pi_1, V_1) \rightarrow (\Pi_2, V_2)\), then \((\{\Pi_1; \Pi\}, V_1) \rightarrow (\{\Pi_2; \Pi\}, V_2)\)

- if \((U,I,V) \models b\), then (if \(b\) \(\Pi_1\) else \(\Pi_2, V) \rightarrow (\Pi_1, V)\)

- if \((U,I,V) \not\models b\), then (if \(b\) \(\Pi_1\) else \(\Pi_2, V) \rightarrow (\Pi_2, V)\)

- \((\text{while } (b) \Pi, V) \rightarrow (\{\text{if } (b) \{\Pi; \text{while } (b) \Pi\}\}, V)\)
SOS-Example

- `(while (a!=0) {c = a; a = b%a; b = c},(a=20, b=12, c=0)) → ...`

\[
\begin{align*}
\text{while (a!=0) } & \{c = a; a = b \% a; b = c\} \quad (a=20, b=12, c=0) \\
\rightarrow & \{\text{if (a!=0)} \times , \text{while (a!=0) } \times 3\} \quad (a=20, b=12, c=0) \\
\rightarrow & \{\ast , \text{while (a!=0) } \times 3\} \quad (a=20, b=12, c=0) \\
\rightarrow & \{c=a; a = b \% a; b = c\} \times 3 \quad (a=20, b=12, c=0) \\
\rightarrow & \{\times , a = b \% a, b = c\} \times 3 \quad (a=20, b=12, c=0) \\
\rightarrow & \{a = b \% a, b = c\} \times 3 \quad (a=20, b=12, c=0) \\
\rightarrow & \{\ast \} \quad (a=20, b=12, c=0) \\
\end{align*}
\]
About operational semantics

• SOS allows to “symbolically execute” a program
• For every $(\Pi_1, V_1)$, there is exactly one sequence $(\Pi_1, V_1) \rightarrow (\Pi_2, V_2) \rightarrow (\Pi_3, V_3) \rightarrow ...$
• Why? What about nondeterminism? Probabilism? Throwing coins, dices, crayons?
Parallelism

- Parallelism is the primary source of nondeterminism in programs
- Thus: hard to understand, error-prone
- But: of increasing importance (multicores)
- in Java, parallelism by multithreading
- key issue: synchronization
  - via shared variables or message passing
  - test-and-set, lock / semaphores, monitors