Computational Logic

Introduction to Model Checking

René Thiemann

Institute of Computer Science
University of Innsbruck

WS 2007/2008

Outline

nanoPromela

- Promela (Process Meta Language) is modeling language for SPIN
  - most widely used model checker SPIN
  - developed by Gerard Holzmann (Bell Labs, NASA JPL)
  - ACM Software Award 2002
- nanoPromela is the core of Promela
  - shared variables and channel-based communication
  - formal semantics of a Promela model is a channel system
  - processes are defined by means of a guarded command language
- No actions, statements describe effect of actions

nanoPromela

nanoPromela-program $\overline{\mathcal{P}} = [\mathcal{P}_1 | \ldots | \mathcal{P}_n]$ with $\mathcal{P}_i$ processes
A process is specified by a statement:

$$
\begin{array}{lll}
\text{stmt} & ::= & \texttt{skip} \mid x := \text{expr} \mid c?x \mid c!\text{expr} \mid \\
& & \text{stmt}_1 ; \text{stmt}_2 \mid \texttt{atomic}\{\text{assignments}\} \mid \\
& & \textbf{if} \;\; :: g_1 \Rightarrow \text{stmt}_1 \;\; \ldots \;\; :: g_n \Rightarrow \text{stmt}_n \;\; \textbf{fi} \mid \\
& & \textbf{do} \;\; :: g_1 \Rightarrow \text{stmt}_1 \;\; \ldots \;\; :: g_n \Rightarrow \text{stmt}_n \;\; \textbf{od} \\
\text{assignments} & ::= & x_1 := \text{expr}_1 ; x_2 := \text{expr}_2 ; \ldots ; x_m := \text{expr}_m
\end{array}
$$

x is a variable in Var, expr an expression and c a channel, g_i a guard

assume the Promela specification is type-consistent

Conditional statements

if $:: g_1 \Rightarrow \text{stmt}_1 \ldots :: g_n \Rightarrow \text{stmt}_n$ **fi**

- Nondeterministic choice between statements stmt_i for which g_i holds
- Test-and-set semantics: (deviation from Promela)
  - guard evaluation + selection of enabled command + execution first atomic step of selected statement is all performed atomically
- The **if**–**fi**–command blocks if no guard holds
  - parallel processes may unblock a process by changing shared variables
  - e.g., when y=0, **if** $:: y > 0 \Rightarrow x := 42$ **fi** waits until y exceeds 0
- Standard abbreviations:
  - **if** g **then** stmt_1 **else** stmt_2 **fi** $\equiv$ **if** $:: g \Rightarrow \text{stmt}_1 :: \neg g \Rightarrow \text{stmt}_2$ **fi**
  - **if** g **then** stmt_1 **fi** $\equiv$ **if** $:: g \Rightarrow \text{stmt}_1 :: \neg g \Rightarrow$ `skip` **fi**

Iteration statements

do $:: g_1 \Rightarrow \text{stmt}_1 \ldots :: g_n \Rightarrow \text{stmt}_n$ **od**

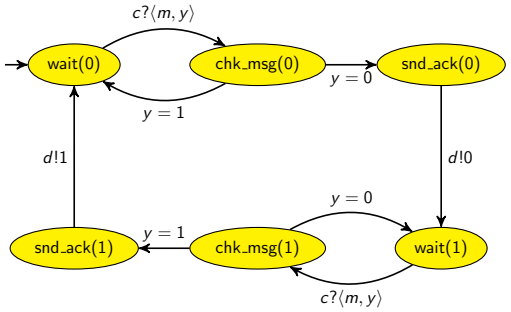
- Iterative execution of nondeterministic choice among $g_i \Rightarrow \text{stmt}_i$
  - where guard g_i holds in the current state
- No blocking if all guards are violated; instead, loop is aborted
- **do** $:: g \Rightarrow$ stmt **od** $\equiv$ **while** g **do** stmt **od**
- No break-statements to abort a loop

Beverage vending machine

The following nanoPromela program describes its behaviour:

do :: true $\Rightarrow$
  `skip;`
  if
    :: $nsprite > 0 \Rightarrow nsprite := nsprite - 1$
    :: $nbeer > 0 \Rightarrow nbeer := nbeer - 1$
    :: $nsprite = nbeer = 0 \Rightarrow$ `skip`
  fi
:: true $\Rightarrow$ `atomic`$\{nbeer := max; nsprite := max\}$
od

Alternating bit protocol - Receiver



Formal semantics

The semantics of a nanoPromela-statement over $(Var, Chan)$ is a program graph over $(Var, Chan)$.

The program graphs $PG_1, \ldots, PG_n$ for the processes $\mathcal{P}_1, \ldots, \mathcal{P}_n$ of a nanoPromela-program $\overline{\mathcal{P}} = [\mathcal{P}_1 | \ldots | \mathcal{P}_n]$ constitute a channel system over $(Var, Chan)$

The locations of the program graph PG_i are the sub-statements of the nanoPromela-program $\mathcal{P}_i$.

Sub-statements

For each statement stmt its substatements $\mathcal{S}ub_{\text{stmt}}(\text{stmt})$ is the smallest set of statements such that

Inference rules

$$\frac{}{\texttt{skip} \xrightarrow{\text{true: } id} \text{exit}}$$

where id denotes an action that does not change the values of the variables

$$\frac{}{x := \text{expr} \xrightarrow{\text{true} \,:\, \text{assign}(x, \text{expr})} \text{exit}}$$

assign(x, expr) denotes the action that only changes x, no other variables

$$\frac{}{c?x \xrightarrow{c?x} \text{exit}} \qquad \frac{}{c!\text{expr} \xrightarrow{c!\text{expr}} \text{exit}}$$

Inference rules

$$\frac{}{\texttt{atomic}\{x_1 := \text{expr}_1; \ldots; x_m := \text{expr}_m\} \xrightarrow{\text{true} \,:\, \alpha_m} \text{exit}}$$

where $\alpha_0 = id$, $\alpha_i = \mathit{Effect}(\text{assign}(x_i, \text{expr}_i), \mathit{Effect}(\alpha_{i-1}, \eta))$ for $1 \leqslant i \leqslant m$

Inference rules

Concurrent programs

- The # states of $P \equiv P_1 \parallel \ldots \parallel P_n$ is maximally:

$$\#\text{states of } P_1 \times \ldots \times \#\text{states of } P_n$$

$\Rightarrow$ # states grows *exponentially* with the number of components

- The composition of N components of size k each yields k^N states
- This is called *the state-space explosion problem*

Sequential programs

- The # states of a simple program graph is:

$$\left|\#\text{program locations}\right| \cdot \prod_{\text{variable } x} |\,dom(x)\,|$$

$\Rightarrow$ number of states grows *exponentially* in the number of program variables

- N variables with k possible values each yields k^N states
- this is called *the state-space explosion problem*

- A program with 10 locations, 3 bools, 4 integers (in range $0 \ldots 9$):

$$10 \cdot 2^3 \cdot 10^4 = 800,000 \text{ states}$$

- Adding a single 50-positions bit-array yields $800,000 \cdot 2^{50}$ states

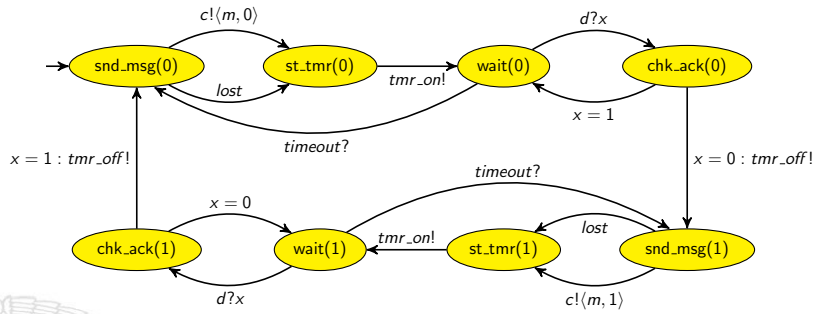
Channel systems

- Asynchronous communication of processes via *channels*
  - each channel c has a bounded capacity $cap(c)$
  - if a channel has capacity 0, we obtain handshaking
- # states of system with N components and K channels is:

$$\prod_{i=1}^{N} \left(\left|\#\text{program locations}\right| \prod_{\text{variable } x} |\,dom(x)\,| \right) \cdot \prod_{j=1}^{K} |dom(c_j)|^{cap(c_j)}$$

this is the underlying structure of Promela

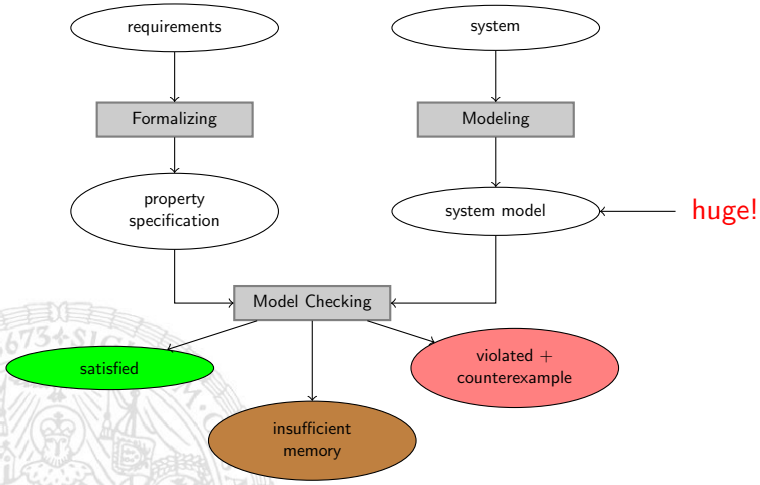
The alternating bit protocol



channel capacity 10, and datums are bits, yields

$$\underbrace{2}_{\text{timer}} \cdot \underbrace{8 \cdot 2}_{\text{sender}} \cdot \underbrace{6 \cdot 2}_{\text{recv}} \cdot \underbrace{4^{10}}_{c} \cdot \underbrace{2^{10}}_{d} = 3 \cdot 2^{37} \approx 4 \cdot 10^{11} \text{ states}$$

The Need for Automated Verification



Summary of Modeling Concurrent Systems

- Transition systems are fundamental for modeling software
- Interleaving = execution of independent concurrent processes by nondeterminism
- Channel systems = program graphs + first-in first-out communication channels
  - handshaking for channels of capacity 0
  - asynchronous message passing when capacity exceeds 0
  - semantical model of Promela
- Formal semantic for Promela ⇒ know exactly which system is verified
- Size of transition systems grows exponentially
  - in the number of concurrent components and the number of variables

Exercises

- Write promela-code for the timer and the sender of the alternating bit protocol.
- Construct the corresponding program graphs according to the formal semantics.
- Change the inference rules such that conditional- and iteration-statements require one step for choosing the case without partially evaluating the corresponding statement.
  (Note that this refinement corresponds to the semantics of full Promela.)