Semantics and Verification of Software

Lecture 18: Nondeterminism and Parallelism I (Shared-Variables and Channel Communication)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH AACHEN UNIVERSITY

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/svsw11/

Winter Semester 2011/12

Studieren ohne Grenzen

Nacht der Professoren

27.01. Apollo 22:00

Ab 23:00 legen eure Professoren von der RWTH auf:

Prof. Fischer | Zahnmedizin
Prof. Lorz | Int. Economics
Prof. Katoen | Informatik
Prof. v.d. Mosel | Mathe
Prof. Bein | Literatur

4€ ab 23h 5€

www.studieren-ohne-grenzen.org

Informatik-Kolloquium
Rheinisch-Westfälische Technische Hochschule Aachen
Lehrstuhl für Informatik 2

EINLADUNG

Zeit:	Mittwoch, 25. Januar 2012, 15:00 Uhr
Ort:	Hörsaal AH 3, Ahornstr. 55
Referent:	Dr. Thomas Noll RWTH Aachen
Thema:	Correctness, Safety and Fault Tolerance in Aerospace Systems: The ESA COMPASS Project

Building modern aerospace systems is highly demanding. They should be extremely dependable, offering service without failures for a very long time – typically years or decades. The need for an integrated system-software co-engineering framework to support the design of such systems is therefore pressing. However, current tools and formalisms tend to be tailored to specific analysis techniques and do not sufficiently cover the full spectrum of required system aspects such as safety, dependability and performability. Additionally, they cannot properly handle the intertwining of hardware and software operation. As such, current engineering practice lacks integration and coherence.

This talk gives an overview of the COMPASS project that was initiated by the European Space Agency to overcome this problem. It supports system-software co-engineering of real-time embedded systems by following a coherent and multidisciplinary approach. We show how such systems and their possible failures can be modeled in the Architecture and Analysis Design Language, how their behavior can be formalized, and how to analyze them by means of model checking and related techniques.

Es laden ein: Die Dozenten der Informatik

Outline

Motivation

- Essential question: what is the meaning of

$$c_1 \parallel c_2$$

(parallel execution of $c_1, c_2 \in Cmd$)?

- Easy to answer when state spaces are disjoint:

$$\langle \texttt{x := 1} \parallel \texttt{y := 2}, \sigma\rangle \to \sigma[\mathrm{x} \mapsto 1, \mathrm{y} \mapsto 2]$$

(no interaction $\Rightarrow$ corresponds to sequential execution)

- But what if variables are shared?

$$(\texttt{x := 1} \parallel \texttt{x := 2}); \texttt{if}\ x = 1\ \texttt{then}\ c_1\ \texttt{else}\ c_2$$

(runs c_1 or c_2 depending on execution order of initial assignments)

- Even more complicated for non-atomic assignments...

Non-Atomic Assignments

Observation: parallelism introduces new phenomena

Example 18.1

$$\begin{array}{c} x := 0; \\ (\underset{13}{x := x + 1} \parallel \underset{2}{x := x + 2}) \end{array} \qquad \text{value of } x\text{: } 0123$$

- At first glance: x is assigned 3
- But: both parallel components could read x before it is written
- Thus: x is assigned 2, 1, or 3
- If exclusive access to shared memory and atomic execution of assignments guaranteed
  $\Rightarrow$ only possible outcome: 3

Parallelism and Interaction

The problem arises due to the combination of

- parallelism and
- interaction (here: via shared memory)

Conclusion

When modeling parallel systems, the precise description of the mechanisms of both parallelism and interaction is crucially important.

Reactive Systems

- Thus: “classical” model for sequential systems

$$\text{System} : \text{Input} \to \text{Output}$$

  (transformational systems) is not adequate
- Missing: aspect of interaction
- Rather: reactive systems which interact with environment and among themselves
- Main interest: not terminating computations but infinite behavior (system maintains ongoing interaction with environment)
- Examples:
  - operating systems
  - embedded systems controlling mechanical or electrical devices (planes, cars, home appliances, ...)
  - power plants, production lines, ...

Overview

Here: study of parallelism in connection with different kinds of interaction

1. Shared-variables communication
2. Channel communication (CSP)
3. Algebraic approaches (CCS)

Outline

The ParWHILE Language

Definition 18.2 (Syntax of ParWHILE)

$$
\begin{array}{l}
a ::= z \mid x \mid a_1\texttt{+}a_2 \mid a_1\texttt{-}a_2 \mid a_1\texttt{*}a_2 \in AExp \\
b ::= t \mid a_1\texttt{=}a_2 \mid a_1\texttt{>}a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in BExp \\
c ::= \texttt{skip} \mid x \texttt{ := } a \mid c_1\texttt{;}c_2 \mid \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2 \mid \texttt{while } b \texttt{ do } c \mid \\
\qquad c_1 \parallel c_2 \in Cmd
\end{array}
$$

- Approach for defining semantics:
  - assignments are executed atomically
  - parallelism is modeled by interleaving, i.e., the actions of parallel statements are merged

$\Rightarrow$ Reduction of parallelism to nondeterminism + sequential execution (similar to multitasking on sequential computers)

- Requires single-step execution relation for statements (cf. Exercise 1.1 for single-step evaluation of expressions)

Semantics of ParWHILE

Definition 18.3 (Single-step execution relation)

The single-step execution relation,

$$\rightarrow_1 \subseteq (Cmd \times \Sigma) \times ((Cmd \times \Sigma) \cup \Sigma),$$

is defined by the following rules:

$$\frac{}{\langle \texttt{skip}, \sigma\rangle \rightarrow_1 \sigma} \qquad \frac{\langle a, \sigma\rangle \rightarrow z}{\langle x := a, \sigma\rangle \rightarrow_1 \sigma[x \mapsto z]}$$

$$\frac{\langle c_1, \sigma\rangle \rightarrow_1 \langle c_1', \sigma'\rangle}{\langle c_1; c_2, \sigma\rangle \rightarrow_1 \langle c_1'; c_2, \sigma'\rangle} \qquad \frac{\langle c_1, \sigma\rangle \rightarrow_1 \sigma'}{\langle c_1; c_2, \sigma\rangle \rightarrow_1 \langle c_2, \sigma'\rangle}$$

$$\frac{\langle b, \sigma\rangle \rightarrow \mathsf{true}}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \rightarrow_1 \langle c_1, \sigma\rangle} \qquad \frac{\langle b, \sigma\rangle \rightarrow \mathsf{false}}{\langle \texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, \sigma\rangle \rightarrow_1 \langle c_2, \sigma\rangle}$$

$$\frac{\langle b, \sigma\rangle \rightarrow \mathsf{true}}{\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \rightarrow_1 \langle c; \texttt{while } b \texttt{ do } c, \sigma\rangle} \qquad \frac{\langle b, \sigma\rangle \rightarrow \mathsf{false}}{\langle \texttt{while } b \texttt{ do } c, \sigma\rangle \rightarrow_1 \sigma}$$

$$\frac{\langle c_1, \sigma\rangle \rightarrow_1 \langle c_1', \sigma'\rangle}{\langle c_1 \parallel c_2, \sigma\rangle \rightarrow_1 \langle c_1' \parallel c_2, \sigma'\rangle} \qquad \frac{\langle c_1, \sigma\rangle \rightarrow_1 \sigma'}{\langle c_1 \parallel c_2, \sigma\rangle \rightarrow_1 \langle c_2, \sigma'\rangle}$$

$$\frac{\langle c_2, \sigma\rangle \rightarrow_1 \langle c_2', \sigma'\rangle}{\langle c_1 \parallel c_2, \sigma\rangle \rightarrow_1 \langle c_1 \parallel c_2', \sigma'\rangle} \qquad \frac{\langle c_2, \sigma\rangle \rightarrow_1 \sigma'}{\langle c_1 \parallel c_2, \sigma\rangle \rightarrow_1 \langle c_1, \sigma'\rangle}$$

Semantics of ParWHILE II

Example 18.4

Let $c := (x := 1 \parallel x := 2); \texttt{if } x = 1 \texttt{ then } c_1 \texttt{ else } c_2$ and $\sigma \in \Sigma$.

$$\langle c, \sigma\rangle \to_1 \langle x := 2; \texttt{if } x = 1 \texttt{ then } c_1 \texttt{ else } c_2, \sigma[x \mapsto 1]\rangle$$

$$\text{since } \frac{\dfrac{\overline{\langle 1, \sigma\rangle \to 1}}{\langle x := 1, \sigma\rangle \to_1 \sigma[x \mapsto 1]}}{\langle x := 1 \parallel x := 2, \sigma\rangle \to_1 \langle x := 2, \sigma[x \mapsto 1]\rangle}$$

$$\to_1 \langle \texttt{if } x = 1 \texttt{ then } c_1 \texttt{ else } c_2, \sigma[x \mapsto 2]\rangle$$

$$\text{since } \frac{\overline{\langle 2, \sigma\rangle \to 2}}{\langle x := 2, \sigma\rangle \to_1 \sigma[x \mapsto 2]}$$

$$\to_1 \langle c_2, \sigma[x \mapsto 2]\rangle$$

$$\text{since } \frac{\overline{\langle x, \sigma[x \mapsto 2]\rangle \to 2} \quad \overline{\langle 1, \sigma[x \mapsto 2]\rangle \to 1}}{\langle x = 1, \sigma[x \mapsto 2]\rangle \to \text{false}}$$

Analogously:

$$\langle c, \sigma\rangle \to_1^3 \langle c_1, \sigma[x \mapsto 1]\rangle$$

Outline

Communicating Sequential Processes

- Approach: Communicating Sequential Processes (CSP) by T. Hoare and R. Milner
- Models system of processors that
  - have (only) local store and
  - run a sequential program ("process")
- Communication proceeds in the following way:
  - processes communicate along channels
  - process can send/receive on a channel if another process simultaneously performs the complementary I/O operation

  $\Rightarrow$ no buffering (synchronous communication)
- New syntactic domains:

Channel names:	$\alpha, \beta, \gamma, \ldots \in \mathit{Chn}$
Input operations:	$\alpha?x$ where $\alpha \in \mathit{Chn}$, $x \in \mathit{Var}$
Output operations:	$\alpha!a$ where $\alpha \in \mathit{Chn}$, $a \in \mathit{AExp}$
Guarded commands:	$gc \in \mathit{GCmd}$

Syntax of CSP

Definition 18.5 (Syntax of CSP)

The syntax of CSP is given by

$$\begin{aligned}
a &::= z \mid x \mid a_1{+}a_2 \mid a_1{-}a_2 \mid a_1{*}a_2 \in AExp\\
b &::= t \mid a_1{=}a_2 \mid a_1{>}a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in BExp\\
c &::= \texttt{skip} \mid x := a \mid \alpha?x \mid \alpha!a \mid\\
&\quad\;\; c_1; c_2 \mid \texttt{if}\ gc\ \texttt{fi} \mid \texttt{do}\ gc\ \texttt{od} \mid c_1 \parallel c_2 \in Cmd\\
gc &::= b \rightarrow c \mid b \wedge \alpha?x \rightarrow c \mid b \wedge \alpha!a \rightarrow c \mid gc_1 \,\Box\, gc_2 \in GCmd
\end{aligned}$$

- In $c_1 \parallel c_2$, statements c_1 and c_2 must not use common variables (only local store)
- Guarded command $gc_1 \,\Box\, gc_2$ represents an alternative
- In $b \rightarrow c$, b acts as a guard that enables the execution of c only if evaluated to true
- $b \wedge \alpha?x \rightarrow c$ and $b \wedge \alpha!a \rightarrow c$ additionally require the respective I/O operation to be enabled
- If none of its alternatives is enabled, a guarded command gc fails (state fail)
- `if` nondeterministically picks an enabled alternative
- A `do` loop is iterated until its body fails

Semantics of CSP I

- Most important aspect: I/O operations
- E.g., $\langle \alpha?x; c, \sigma\rangle$ can only execute if a parallel statement provides corresponding output

$\Rightarrow$ Indicate communication potential by labels

$$L = \{\alpha?z \mid \alpha \in Chn, z \in \mathbb{Z}\} \cup \{\alpha!z \mid \alpha \in Chn, z \in \mathbb{Z}\}$$

- Yields following labeled transitions:

$$\langle \alpha?x; c, \sigma\rangle \xrightarrow{\alpha?z} \langle c, \sigma[x \mapsto z]\rangle \quad (\text{for all } z \in \mathbb{Z})$$
$$\langle \alpha!a; c', \sigma\rangle \xrightarrow{\alpha!z} \langle c', \sigma\rangle \quad (\text{if } \langle a, \sigma\rangle \to z)$$

- Now both statements, if running in parallel, can communicate:

$$\langle (\alpha?x; c) \parallel (\alpha!a; c'), \sigma\rangle \to \langle c \parallel c', \sigma[x \mapsto z]\rangle.$$

- To allow communication with other processes, the following transitions should also be possible (for all $z' \in \mathbb{Z}$):

$$\langle (\alpha?x; c) \parallel (\alpha!a; c'), \sigma\rangle \xrightarrow{\alpha?z'} \langle c \parallel (\alpha!a; c'), \sigma[x \mapsto z']\rangle$$
$$\langle (\alpha?x; c) \parallel (\alpha!a; c'), \sigma\rangle \xrightarrow{\alpha!z} \langle (\alpha?x; c) \parallel c', \sigma\rangle$$

Semantics of CSP II

Definition of transition relation

$$\xrightarrow{\lambda} \subseteq (Cmd \times \Sigma) \times (Cmd \times \Sigma) \cup (GCmd \times \Sigma) \times (Cmd \times \Sigma \cup \{\mathsf{fail}\})$$

(see following slides)

- Marking λ can be a label or empty: $\lambda \in L \cup \{\varepsilon\}$
- Uniform treatment of configurations of the form $\langle c, \sigma \rangle \in Cmd \times \Sigma$ and $\sigma \in \Sigma$:
    - σ interpreted as $\langle *, \sigma \rangle$ with "empty" statement $*$
    - $*$ satisfies $*; c = c; * = * \parallel c = c \parallel * = c$
- Thus: read $\langle x := 0 \parallel *, \sigma \rangle$ as $\langle x := 0, \sigma \rangle$

Semantics of CSP III

Definition 18.6 (Semantics of CSP)

Rules for statements

$$\frac{}{\langle \texttt{skip}, \sigma\rangle \to \sigma} \qquad \frac{\langle a, \sigma\rangle \to z}{\langle x := a, \sigma\rangle \to \sigma[x \mapsto z]}$$

$$\frac{}{\langle \alpha?x, \sigma\rangle \xrightarrow{\alpha?z} \sigma[x \mapsto z]} \qquad \frac{\langle a, \sigma\rangle \to z}{\langle \alpha!a, \sigma\rangle \xrightarrow{\alpha!z} \sigma}$$

$$\frac{\langle c_1, \sigma\rangle \xrightarrow{\lambda} \langle c_1', \sigma'\rangle}{\langle c_1; c_2, \sigma\rangle \xrightarrow{\lambda} \langle c_1'; c_2, \sigma'\rangle} \qquad \frac{\langle gc, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle}{\langle \texttt{if}\ gc\ \texttt{fi}, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle}$$

$$\frac{\langle gc, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle}{\langle \texttt{do}\ gc\ \texttt{od}, \sigma\rangle \xrightarrow{\lambda} \langle c; \texttt{do}\ gc\ \texttt{od}, \sigma'\rangle} \qquad \frac{\langle gc, \sigma\rangle \to \text{fail}}{\langle \texttt{do}\ gc\ \texttt{od}, \sigma\rangle \to \sigma}$$

$$\frac{\langle c_1, \sigma\rangle \xrightarrow{\lambda} \langle c_1', \sigma'\rangle}{\langle c_1 \parallel c_2, \sigma\rangle \xrightarrow{\lambda} \langle c_1' \parallel c_2, \sigma'\rangle} \qquad \frac{\langle c_2, \sigma\rangle \xrightarrow{\lambda} \langle c_2', \sigma'\rangle}{\langle c_1 \parallel c_2, \sigma\rangle \xrightarrow{\lambda} \langle c_1 \parallel c_2', \sigma'\rangle}$$

$$\frac{\langle c_1, \sigma\rangle \xrightarrow{\alpha?z} \langle c_1', \sigma'\rangle, \langle c_2, \sigma\rangle \xrightarrow{\alpha!z} \langle c_2', \sigma\rangle}{\langle c_1 \parallel c_2, \sigma\rangle \to \langle c_1' \parallel c_2', \sigma'\rangle} \qquad \frac{\langle c_1, \sigma\rangle \xrightarrow{\alpha!z} \langle c_1', \sigma\rangle, \langle c_2, \sigma\rangle \xrightarrow{\alpha?z} \langle c_2', \sigma'\rangle}{\langle c_1 \parallel c_2, \sigma\rangle \to \langle c_1' \parallel c_2', \sigma'\rangle}$$

Semantics of CSP IV

Definition 18.6 (Semantics of CSP; continued)

Rules for guarded commands:

$$\frac{\langle b, \sigma\rangle \to \mathsf{true}}{\langle b \to c, \sigma\rangle \to \langle c, \sigma\rangle} \qquad \frac{\langle b, \sigma\rangle \to \mathsf{false}}{\langle b \to c, \sigma\rangle \to \mathsf{fail}}$$

$$\frac{\langle b, \sigma\rangle \to \mathsf{true}}{\langle b \wedge \alpha?x \to c, \sigma\rangle \xrightarrow{\alpha?z} \langle c, \sigma[x \mapsto z]\rangle} \qquad \frac{\langle b, \sigma\rangle \to \mathsf{false}}{\langle b \wedge \alpha?x \to c, \sigma\rangle \to \mathsf{fail}}$$

$$\frac{\langle b, \sigma\rangle \to \mathsf{true}, \langle a, \sigma\rangle \to z}{\langle b \wedge \alpha!a \to c, \sigma\rangle \xrightarrow{\alpha!z} \langle c, \sigma\rangle} \qquad \frac{\langle b, \sigma\rangle \to \mathsf{false}}{\langle b \wedge \alpha!a \to c, \sigma\rangle \to \mathsf{fail}}$$

$$\frac{\langle gc_1, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle}{\langle gc_1 \,\square\, gc_2, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle} \qquad \frac{\langle gc_2, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle}{\langle gc_1 \,\square\, gc_2, \sigma\rangle \xrightarrow{\lambda} \langle c, \sigma'\rangle}$$

$$\frac{\langle gc_1, \sigma\rangle \to \mathsf{fail}, \langle gc_2, \sigma\rangle \to \mathsf{fail}}{\langle gc_1 \,\square\, gc_2, \sigma\rangle \to \mathsf{fail}}$$

Semantics of CSP V

Example 18.7

1. $\texttt{do}\ (\mathsf{true} \wedge \alpha?x \to \beta!x)\ \texttt{od}$
   describes a process that repeatedly receives a value along α and forwards it along β
   (reception and forwarding of value 1: on the board)
2. $\texttt{do}\ \mathsf{true} \wedge \alpha?x \to \beta!x\ \texttt{od} \parallel \texttt{do}\ \mathsf{true} \wedge \beta?y \to \gamma!y\ \texttt{od}$
   specifies a buffer of capacity 2 that receives along α and sends along γ (using β for internal communication)
3. Nondeterministic choice between input channels:
   1. $\texttt{if}\ (\mathsf{true} \wedge \alpha?x \to c_1 \mathbin{\Box} \mathsf{true} \wedge \beta?y \to c_2)\ \texttt{fi}$
   2. $\texttt{if}\ (\mathsf{true} \to (\alpha?x; c_1) \mathbin{\Box} \mathsf{true} \to (\beta?y; c_2))\ \texttt{fi}$

   Expected: progress whenever environment provides data on α or β
   1. correct
   2. incorrect (can deadlock – on the board)