Towards an executable algebra for product lines

STEFANIA GNESI, MARINELLA PETROCCHI

PRESENTER: MAURICE TER BEEK

NATIONAL RESEARCH COUNCIL
PISA, ITALY

FMSPLE 2012
SEPTEMBER 2, 2012
Outline

- Variability through an example
- Our approach and our goal
- Specification language: CL4SPL
- Verification toolkit: Maude
- Discussion
A family of coffee machines

- Initially, a coin must be inserted
  - Euros for European machines
  - Dollars for Canadian machines

- Then, either sugar can be selected, or directly a beverage can be selected

- Beverages are coffee, tea, and cappuccino
  - All machines must serve coffee, only European machines can serve cappuccino, tea can be served by both machines

- A ringtone can be rung after beverage delivery
  - The ringtone must be rung after cappuccino delivery

- Finally, the machine returns idle
Variability aspects in PFs

- **Optional features**: either can be present or not in some products of the family:
  - Tea selection

- **Mandatory features**: must be present in all products of the family:
  - Coffee selection

- **Alternative features**: only one of them can be executed:
  - Either dollar or euro can be inserted

- **One feature requires/excludes others**:
  - Ringtone rung after cappuccino is served
  - No cappuccino in Canadian machines
Our approach

- All products in a family are represented by a triple:
  - A set of fragments
    - Fragment = <subject, action, object>
    - Example: <user, select, coffee>
  - A set of contexts
    - Contexts are propositional formulas over an assertion
    - Example: comes-from(machine, Europe)
    - Either true or false
  - An algebraic process over fragments (as actions of the process) and over contexts (as tests in conditional expressions)

- A product configuration is selected by setting some contexts to TRUE
Our goal

- Intuitive representation of PFs through a controlled natural language
- Automated generation of valid products
- Verification of both safety and liveness properties over products
- All this in a well-known and rich toolkit, with many more possibilities than those explored for now (e.g. theorem proving, quantitative model checking)
Based on contexts \( c \), fragments \( f \), and processes \( P \)

Basic and composable contexts:

\[
C = c \mid C \land C \mid C \lor C \mid \text{not } c
\]

Fragments:

\[
f = \langle s, a, o \rangle \quad \text{“subject } s \text{ performs action } a \text{ on object } o \text{”}
\]

Processes:

\[
P = \emptyset \mid A \mid \text{after } f \text{ then } P \mid \text{if } c \text{ then } P_1 \text{ else } P_2 \mid P_1 \text{ or } P_2 \mid (P)
\]
Example specification

\[ \text{CMACHINE} \equiv (\text{after } f_{\text{start}} \text{ then } P_1) \text{ or CMACHINE} \]

\[ P_1 \equiv \text{if } c_{\text{europe}} \text{ then } (\text{after } f_{\text{euro}} \text{ then } (P_2 \text{ or } P_3) \text{ else } (\text{after } f_{\text{dollar}} \text{ then } (P_2 \text{ or } P_3))) \]

\[ P_2 \equiv \text{after } f_{\text{sugar}} \text{ then } P_3 \]

\[ P_3 \equiv (\text{after } f_{\text{coffee}} \text{ then } P_4 \text{ else } P_3) \text{ or } (\text{if } c_{\text{tea}} \text{ then } (\text{after } f_{\text{tea}} \text{ then } P_4) \text{ else } P_3) \text{ or } (\text{if } (c_{\text{europe}} \land c_{\text{cap}}) \text{ then } (\text{after } f_{\text{cap}} \text{ then } P_5) \text{ else } P_3) \]

\[ P_4 \equiv P_5 \text{ or CMACHINE} \]

\[ P_5 \equiv \text{after } f_{\text{ring}} \text{ then } CMACHINE \]
Maude

- http://maude.cs.uiuc.edu
- Specification language based on Rewriting Logic
- Distributed systems specified as:
  - Algebraic data types axiomatizing system states
  - Rewriting rules axiomatizing system transitions
- Executable programming language
- Comes with a rich toolkit that allows formal reasoning on the produced specification, like (real-time, probabilistic) model checking, theorem proving, etc.
Maude modules

- A collection of sorts and operations on them
- The information to reduce and rewrite input expressions of the Maude environment
- Functional modules define equations
- System modules map system transitions into rewrite rules
- Example of a system module:
  - mod climate is
  - sort weathercondition .
  - op sunnyday : -> weathercondition .
  - op rainyday : -> weathercondition .
  - endm
From CL4SPL to Maude

- CL4SPL has a formal foundation based on labelled transition systems, thus allowing for a translation to rewriting logic-based languages.

\[
\begin{align*}
\text{(after)} & \quad \text{after } f \text{ then } P \xrightarrow{f} P \\
\text{(if)} & \quad \text{if } C \text{ then } P_1 \xrightarrow{f} P'_1 \\
\text{(or)} & \quad \text{if } C \text{ then } P_1 \text{ else } P_2 \xrightarrow{f} P'_1 \\
\text{(or)} & \quad \text{if } C \text{ then } P_1 \text{ else } P_2 \xrightarrow{f} P'_1
\end{align*}
\]

- We implemented an executable specification of CL4SPL into Maude.
Example analysis (1)

- Generation of valid products
- With the following contexts set to true:
  - comes-from(machine, canada) = true;
  - offers(machine, coffee) = true;
  - offers(machine, tea) = true;

the process CMACHINE can be rewritten into a number of possible solutions ("search" command)

- This allows to explore the reachable state space and visualize the resulting traces (PF behaviour)
Product behaviour

- Two possible ways of rewriting a Canadian CMACHINE
- 19: start the machine, insert a dollar, press sugar button, select a coffee, back to initial state without ringtone, starts again, this time selecting a tea, without sugar, etc.

Solution 18 (state 18)
states: 19  rewrites: 1050 in 4ms cpu (3ms real) (262500 rewrites/second)
FP: FProcess --> {< 'user,'start,'machine '>}{< 'user,'insert,'dollar '>}{< 'user,'press,'sugar '>}{< 'user,'select,'coffee '>}{< 'user,'start,'machine '>}{< 'user,'insert,'dollar '>}{< 'user,'select,'coffee '>}'P4

Solution 19 (state 19)
states: 20  rewrites: 1064 in 4ms cpu (3ms real) (266000 rewrites/second)
FP: FProcess --> {< 'user,'start,'machine '>}{< 'user,'insert,'dollar '>}{< 'user,'press,'sugar '>}{< 'user,'select,'coffee '>}{< 'user,'start,'machine '>}{< 'user,'insert,'dollar '>}{< 'user,'select,'tea '>}'P4
Specific capabilities queries ("red(uce)" command)

For instance, after that the user starts the machine:

- It is not possible to select a coffee if no money has been inserted before (TRUE for both European and Canadian machines);
- After the user inserts one dollar (s)he can select either a coffee or a tea, but not a cappuccino (TRUE for Canadian machines only);
- After the user inserts a coin, then s(he) can select a beverage (TRUE for both European and Canadian machines);
- After the user inserts one euro and selects a cappuccino, then a ringtone must follow (TRUE for European machines only)
Capabilities of contextualized coffee machine

- After selecting a cappuccino, it is not possible to get back to the initial state without ringing a ringtone.
CL4SPL can deal with variability aspects

Optional/alternative features and excludes relation: appropriately set the truth values of contexts

- Either tea can be selected or not, depending on the value of context “offers(machine, tea)"
- Is machine Canadian? This enables the alternative “insert dollar”
- Is machine Canadian? This excludes to serve “cappuccino”

Mandatory features: no guard in front of the fragment

- No guard in front of coffee always allows to serve coffee

Requires relation: mainly by temporal sequentialization

- After a cappuccino is served, then a ringtone is required
Some future directions

- Variability aspects, like requires/excludes relations, may be inserted directly as constructs of the language.

- Specification and analysis of temporal and probabilistic properties of PFs:
  - Is a beverage always served within a certain amount of time?
  - Which is the probability that the beverage will be served, based on the last time that the beverage has been loaded?

- Enrichment through dynamic contexts:
  - A coffee machine will serve cappuccino iff milk is available.
Publicity: submit to VaMoS 2013 in Pisa!

Variability Modelling of Software-Intensive Systems (VaMoS'13)
7th International Workshop
Pisa, Italy, January 23--25, 2013

- http://www.vamos-workshop.net
- **Submission deadline: November 4, 2012**

PC chairs:
- Philippe Collet (Université Nice Sophia Antipolis, France)
- Klaus Schmid (Stiftung Universität Hildesheim, Germany)

- Organized by our FMT lab at the CNR research area in Pisa