Implementation of a compiler from Pluscal to TLA+ with Tom

Marc PINHEDE

ESIAL-Telecom Nancy
Contents

I Environement
   • Pluscal2.0
   • TLA+
   • Tools

II Personal work
   • Grammars
   • Tree rewriting
   • Code Generation

III Situation after the internship
   • Tests
   • Todo
Introduction

- Location: Loria
- Team: AlGorille
- Supervisor: Martin Quinson
- Task: Rework on a thesis compiler
- Subject: Compiler form Pluscal2.0 to TLA+
Introduction

Situation:

TLA+, a language used to specify a system:

- Very mathematical specification
- Permit to use a model-checker
- But not an easy language to learn, for program designers.

Leslie Lamport introduced Pluscal. But it was still not really easy to use.
Pluscal 2.0

- Syntax close to standard algorithms
  - Procedures can be used
  - Non typed variables
- Accept processes and hierarchical processes
- Atomicity for some part of code accepted
- Embedded TLA+ code
algorithm Peterson
extends Naturals
constants
  numPeers (* Number of processes *)
variables
  lockReq = [id ∊ Node |→ FALSE],
  turn = 1, (* tie-break variable *)
  count = 0 (* number of processes holding the lock *)

fair process Node[numPeers]
definition other == CHOOSE id ∊ Node : id # self
begin
ncs: loop
  skip;
  lockReq[self] := TRUE;
  turn := other;
try:
  when ~lockReq[other] \ turn = self;
cs:
  count := count + 1;
leave:
  count := count - 1;
  lockReq[self] := FALSE;
end loop;
end process;

(*No Main process*)

(* Assert: at most one process have the lock *)
invariant count <= 1

(* Liveness: each requested lock is eventually granted *)
temporal \( \forall p \in \text{Node}: [] (<> \text{lockReq}[p]) \)

(* Instantiating the model for 2 processes *)
constants numPeers = 2
Set of Actions

Action:
- Guard conditions
- Variable modifications
- List of unchanged variables

Special actions that ensure liveness properties
--- MODULE HourClock ---

EXTENDS Naturals

VARIABLE hr

HCini == hr ∈ (1..12)
HCnxt == hr’ = IF hr ≠ 12 THEN hr+1 ELSE 1
HC == HCini ∧ []HCnxt_hr

THEOREM HC => []HCini

Vrai == hr ≠ 13
Faux == hr ≠ 7
Main tools:

- Tom: Language extension. Permit easy tree manipulations.
- Antlr: Automatic Parser/Lexer generator.
- Others: Text editor, TlaToolbox, Tlc...
Grammars

Grammars are divided in two set: one for Pluscal2.0, the other for TLA+. For each set, two grammars. The antlr version and the Tom version.

Four files:

- Antlr grammar for Pluscal2.0
- Antlr grammar for TLA+
- Gom signature for Pluscal2.0
- Gom signature for TLA+
Tree rewriting

Tree rewriting are executed by tom.
Two tools:
- match: Applied on a piece of the tree.
- strategy: Applied on the whole tree.

Exemple:

visit LabeledStatement{
    /* Add label to any loop statement.*/
    Labellisation(EmptyLabel(),Loop(labeledStatementList))->
    {
        return 'Labellisation(GivenLabel(OptionString("loop",
            ConcOption())),Loop(labeledStatementList));
    }
}
Rewriting steps

- Arbre Tom Pluscal 2.0
- Semantic Control
- Normalization
- Program flow management
- Translation

Arbre tom TLA+

Terminé pour la version actuelle
Incomplet pour la version actuelle
Éventuel ajout
Code generation

- Use of a 'pretty printer'
- Recursive walk of the tree
- Two output files (.tla et .cfg)
- Completed step for the actual TLA+ signature
Tests

- Approach close to the TDD (Test-Driven Development) used.

- Set of tests, divided in subset, available
  Tests identify working instructions.

- Scripted tests with recorded answers.
Test of the whole compiler

Implementation of a simple test requiring every steps of the compiler

Test accepted by the model-checker

Verification of trivial properties validated
Unfinished parts

- Semantic control
- Normalization for some instructions
- Potential add of a separate step to manage the PC value or Translator completion.
- Add of new features to the compiler
Conclusion

For the compiler:

- Use of traditional tools and separation of the work in steps to make sources more accessible
- Completed main process
- Some steps need to be completed or extended.

As a personal experience:

- Rewarding internship and good approach of the research world
- New tools and new way to program discovered
- Management and teamwork pleasant
Sources

Vérification Formelle d’Algorithmes Distribués en PlusCal-2 - Sabina AKHTAR

Specifying Systems - Leslie Lamport

A Pluscal User’s Manual - Leslie Lamport
Thanks to:

Martin QUINSON, my internship supervisor, for his presence and control, but also for the good atmosphere he was able to create in the team.

Stephan MERTZ, for helping me to understand the potential of this project.

Jean-Christophe BACH, for his help and patience with Tom.