SDL-2000 Tutorial

Prof. J. Fischer
Dr. E. Holz
Dr. A. Prinz

Session C

Humboldt-Universität zu Berlin

Contents

- formal semantics definitions
  - reasons
  - methods
- SDL-2000 formal semantics
  - overview
  - introduction to Abstract State Machines (ASM)
  - selected details of the semantics
  - implementation

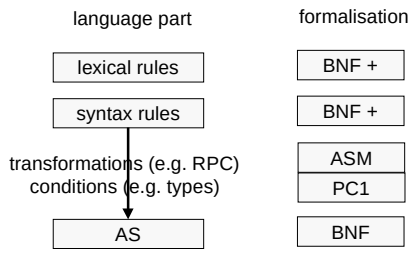
Formal Semantics Definition: Reasons

- avoid ambiguity
- better language understanding
- language properties (e.g. type safety)
- analysis (e.g. model checking)
- some objectives
  - intelligibility
  - maintainability
  - tool support

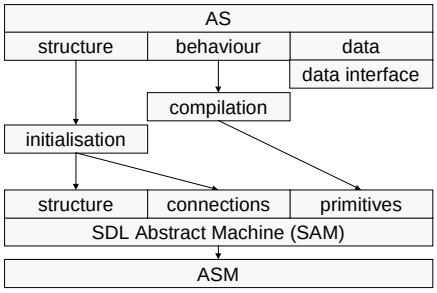
Formal Semantics Definition: Methods

- axiomatic
  - objects and relations between them
- denotational
  - abstract compilation
- operational
  - abstract interpretation

Formality: Static Part



Formality: Dynamic Part



Abstract State Machines

- general
  - state transitions
  - introduced by Gurevich
  - based on mathematics
- states
- transitions
- programs
- concurrency/time

Y. Gurevich. Evolving Algebra 1993: Lipari Guide
In E. Börger, editor, Specification and Validation Methods; Oxford University Press 1995

Y. Gurevich.
ASM Guide 97
CSE Technical Report, University of Michigan-Ann Arbor, 1997

Abstract State Machines

- general
- states
  - algebras
  - sets,functions
  - domains
  - reserve elements
- transitions
- programs
- concurrency/time

vocabulary	state
0-ary function	element
1-ary predicate	domain
function name	function
predicate name	boolean function

Abstract State Machines

- general
- state
- transitions
  - locations, updates
  - firing of updates
  - runs
- programs
- concurrency/time

Account(Jill):= 1000000
Jill.Account:= 1000000

Account(Jack):= 0
Jack.Account:= 0

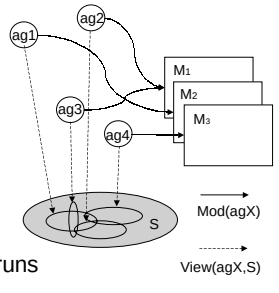
Abstract State Machines

- general
- states
- transitions
- programs
  - do-forall
  - choose
  - if-then-else
  - extend
- concurrency/time

```
if strike then
  extend Flyer by f
else
  do forall s: Student
    know(s):= know(s)+SDL
  choose s: Student
    focus(Self):= s
```


Abstract State Machines

- general
- states
- transitions
- programs
- concurrency/time
  - agents
  - Mod, Self
  - partially ordered runs
  - now



Details: SAM

```
Link_Module
if Self.from.queue<>empty then
  let S=Self.from.queue.head in
    if Applicable(Self,S) then
      DELETE(S,Self.from)
      INSERT(S,Delay(Self),Self.to)
      Self.last_time:=Delay(Self)
where
    Applicable(Self,S)= ...
    Delay(Self)=max(now+Self.delay,Self.last_time)
```

Details: Initialisation

```
Init_Agent_Module
if mode(Self)=initial then
  mode(Self):= starting
  CREATEVARIABLES(Self.ref.Variable-definition)
  do forall a:Self.ref.Agent
    extend Agent with ag
      ref(ag):= a,      mode(ag):= initial
      Mod(ag):= Init_Agent_Module
else
  CREATECHANNELS(Self.ref.Channel-definition)
  Mod(Self):= Execute_Module
```


Details: Compilation

l_0: OUTPUT(gameid, user) => l_1
l_1: STATENODE(/*none*/ l_n,
{<probe, l_2>, <result, l_r>, <endGame, l_e>})
l_2: OUTPUT(lose, user) => l_3
l_3: ASSIGN(count, count-1) => l_1
...

Implementation

tools used: lex, yacc, kimwitu, make, workbench

lexical structure	sdl.l
concrete syntax	sdl.y, sdl_cs.k
abstract syntax	sdl_as.k
transformations	sdl_trans.k
conditions	sdl_cond.k
mapping CS -> AS	sdl_map.k
compilation	sdl_compile.k
ASM parts	sdl.asm

SDL-2000 Tutorial

Prof. J. Fischer
Dr. E. Holz
Dr. A. Prinz

End of Session C

Humboldt-Universität zu Berlin

Further Information

- tutorial slides and author contact
  www.informatik.hu-berlin.de/Institut/struktur/systemanalyse/
  {fischer|holz}@informatik.hu-berlin.de
  prinz@DResearch.de
- ITU standards and recommendations
  - www.itu.ch or www.itu.int/itudoc/itu-t/approved/z/index.html
- SDL Forum Society
  - www.sdl-forum.org
- conferences and workshops
  - bi-annual SDL Forum - next Copenhagen 2001
  - bi-annual Workshop on SDL and MSC (SAM)
    SAM-2000: Grenoble/France, June, 26-28th 2000
    http://www.irisa.fr/manifestations/2000/sam2000/