Using KIV to Verify ASBRU Plans – Overview

Augsburg, 28.11.2001, Michael Balser

Contents

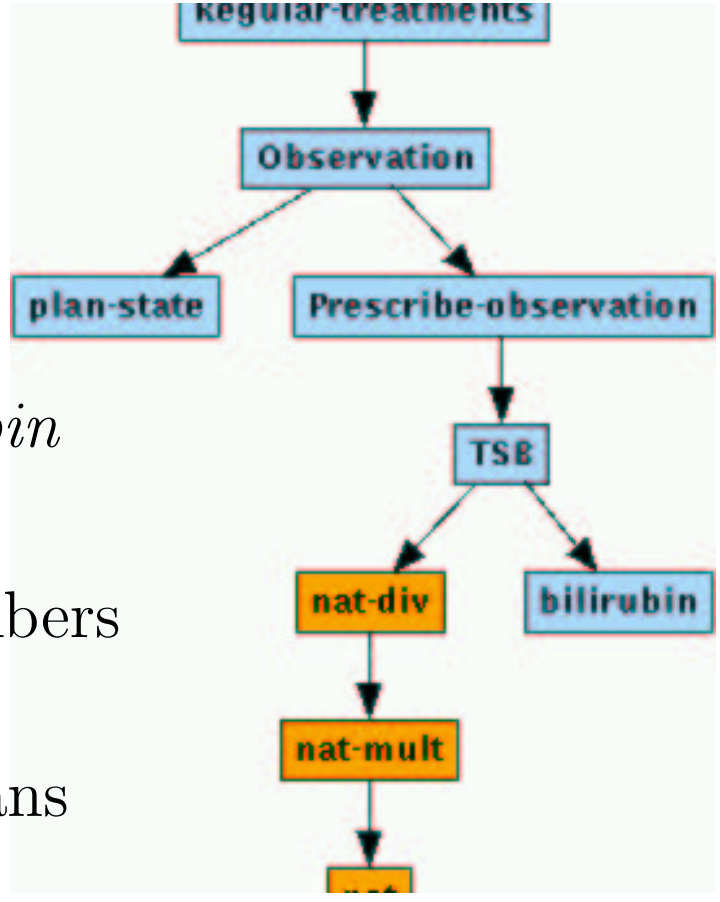
1. ASBRU Example
2. Specification
   (a) Data Abstraction
   (b) ASBRU Plan
   (c) Properties
3. Verification
   (a) Principle
   (b) Symbolic Execution
   (c) Induction
   (d) Modular Proofs
   (e) Automation

Example – ASBRU Plan

PLAN	Observation (1.4.2.1.5)
PREFERENCES	
INTENTIONS	`MAINTAIN INTERMEDIATE STATE` `bilirubin = observation`
CONDITIONS	**Filter:** `(bilirubin observation` `* [_,_] [0,_] [_,_] NOW)` **Abort:** `(bilirubin not observation` `* [_,_] [0,_] [_,_] NOW)`
EFFECTS	
PLAN BODY	**Prescribe-Observation**
COMMENTS	…

Specification – Overview

$\Rightarrow$ *Development graph*



- Data abstraction
  - ⋆ Qualitative bilirubin levels $\Rightarrow$ *bilirubin*
  - ⋆ Abstraction function $\Rightarrow$ *TSB*
- Quantitative TSB levels as natural numbers
  $\Rightarrow$ *library*
- Hierarchical specification of ASBRU plans
  $\Rightarrow$ *modularity*

Specification – Data Abstraction

- Abstract values of bilirubin
  $\Rightarrow$ Enumeration type *bilirubin*
- Abstraction function
  $\Rightarrow$ Function *get-bilirubin* in *TSB*

Abstraction table

age	TSB level	
	-12	...
⋮	⋮	...
25-48	observation	...
⋮	...	...

Predicate logic

$$25 \leq a \wedge a \leq 48 \wedge l < 12$$
$$\rightarrow \text{ get-bilirubin}(\text{set-age-level}(tsb, a, l))$$
$$= \text{observation}$$

Specification – ASBRU Plan

$\Rightarrow$ Specification *Observation*

- Parallel program
- Plan as procedure
- Filter precondition as **await** statement
- Abort condition as **break** statement
- Simple plan body

Atomic plan

$\Rightarrow$ Specification *Prescribe-observation*

- Duration is nondeterministic
- Manual activation as nondeterminism

Specification – Properties

- Termination

$$\begin{array}{lll} & \text{Observation\#}(;\text{tsb},\text{time},\text{Observation-state}), & \textit{System} \\ & \Box\ \text{time}'' = \text{time}' + 1 & \textit{Environment} \\ \vdash & \Box\ \text{get-bilirubin}(\text{tsb}) = \text{observation} \rightarrow \Diamond\ \textbf{last}; & \textit{Property} \end{array}$$

- Intention

$$\begin{array}{lll} & \text{Observation\#}(;\text{tsb},\text{time},\text{Observation-state}), & \textit{System} \\ & \text{Observation-state} = \text{considered} & \textit{Environment} \\ \vdash & \Box \quad \neg\ \textbf{last} \wedge \text{Observation-state} = \text{activate} & \textit{Property} \\ & \quad \rightarrow \text{get-bilirubin}(\text{tsb}) = \text{observation}; & \end{array}$$

$\Rightarrow$ Theorems in specification *Observation*

Specification – General Remarks

State

- Algebraic specifications
- Higher order logic
- Library of standard data types

Transition system

- Parallel programs
- Special ASBRU constructs
- Temporal logic

Important: modular structure

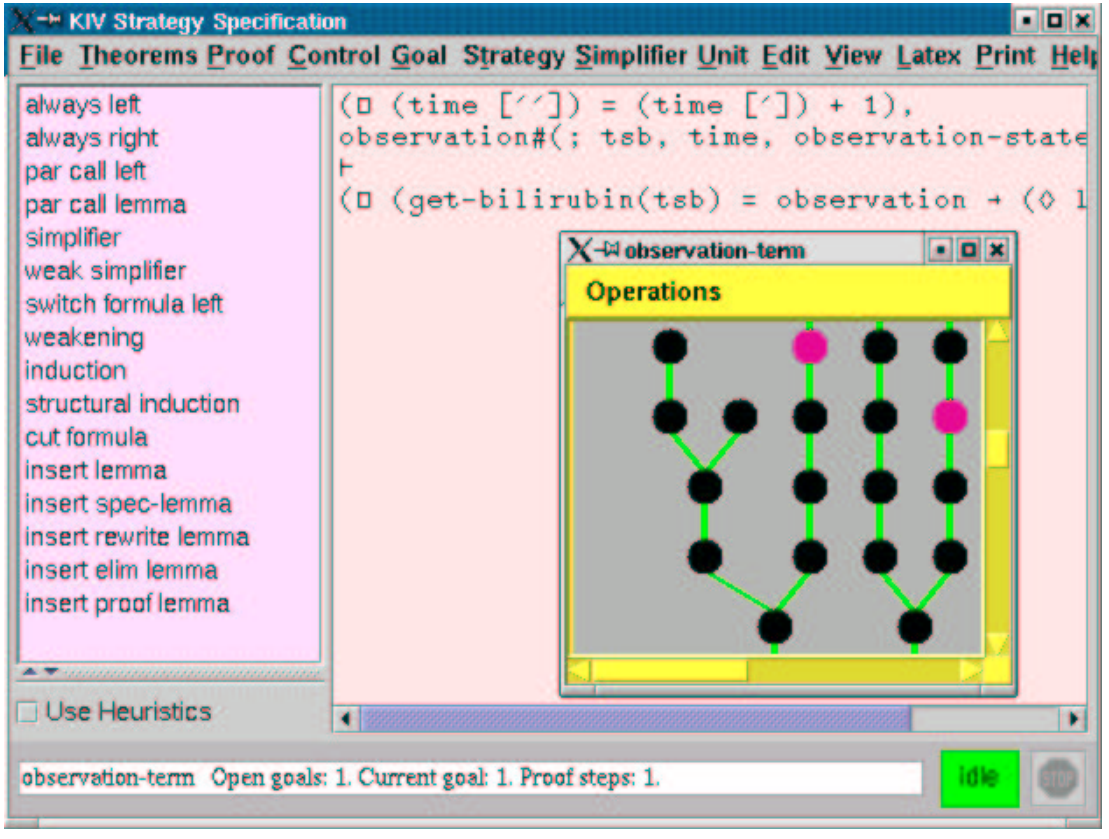
Verification

First step

Construct proof for *Prescribe-observation*
$\Rightarrow$ Prove lemma *prescribe-observation-term*

KIV environment

- Interactive verification
- Choose from set of applicable proof rules
- Proof tree represents progress



Verification – Principle

Proof principle

Symbolic Execution with a Little Induction

- Symbolic Execution
  1. Prove property for current state
  2. Execute one step of ASBRU plan
  3. Continue with next step
- Trace Induction

Modular Proofs

Assumption-Guarantee Proofs

Verification – Symbolic Execution

1. Prove property $\Diamond$ **last**
   $\Rightarrow$ *eventually right*
2. call procedure
   $\Rightarrow$ *par call left*
3. Execute first statement **skip**
   $\Rightarrow$ *cl* $\ll$*rule*$\gg$, *par chop left*
4. Apply environment assumption
   $\Rightarrow$ *always left*
5. Continue with next step
   $\Rightarrow$ *step*, *simplifier*

Prescribe-observation#(...),
$\Box$ Env $\vdash \Diamond$ **last**

skip; **var** ... **in await** ...,
$\Box$ Env $\vdash$ **last**, $\circ \Diamond$ **last**

var ... **in await** ...,
$\Box$ Env, PL $\vdash \Diamond$ **last**

Verification – Induction

Next statement is **await**, which may loop

loc . . . . **await** . . . ,

□ Env, PL ⊢ ◇ **last**

1. Generalisation
   ⇒ *weakening*
2. Induction
   ⇒ *trace induction*

loc . . . . l :- **await** . . . ,

□ Env, IndHyp ⊢ ◇ **last**

3. Execution of next step
   ⇒ . . .

loc . . . . l :- **await** . . . ,

□ Env, IndHyp, PL ⊢ ◇ **last**

4. ⇒ *apply trace induction*

Verification – Modular Proofs

await $\ldots;\ldots;$

break Prescribe-observation#$(\ldots);\ldots$ **if** $\ldots,$

$\ldots \vdash \Diamond$ **last**

1. Execute plan until call to sub plan is reached

   break Prescribe-observation#$(\ldots);\ldots$ **if** $\ldots,\ldots \vdash \Diamond$ **last**

2. Use lemma for sub plan
   $\Rightarrow$ *par call lemma*
3. Prove assumption of lemma

   break $\neg \Box\ time'' = time' + 1;\ldots$ **if** $\ldots,\ldots \vdash \Diamond$ **last**

4. Use guarantee of lemma

   break $\Diamond$ **last**$;\ldots$ **if** $\ldots,\ldots \vdash \Diamond$ **last**

Verification – Automation

Heuristics

Heuristics apply rules automatically

- Symbolic execution can be automated
- Induction can be partly automated

$\Rightarrow$ *Configure Heuristics*

Degree of Automation

- Work in progress
- Proofs fully automatic except
  - $\star$ generalisation and
  - $\star$ modular proofs

Verification – General Remarks

Features of KIV

- Strong proof support for higher order logic
- Automatic simplification
- High degree of automation
- Lemma administration
  - ⋆ Dependency analysis
  - ⋆ Change management
  - ⋆ Reuse of proofs
- Proof manipulation
  - ⋆ Graphical proof trees
  - ⋆ Pruning, backtracking
  - ⋆ Context sensitive rule selection