



Using KIV to Verify ASBRU Plans – Overview



Augsburg, 28.11.2001, Michael Balsler

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



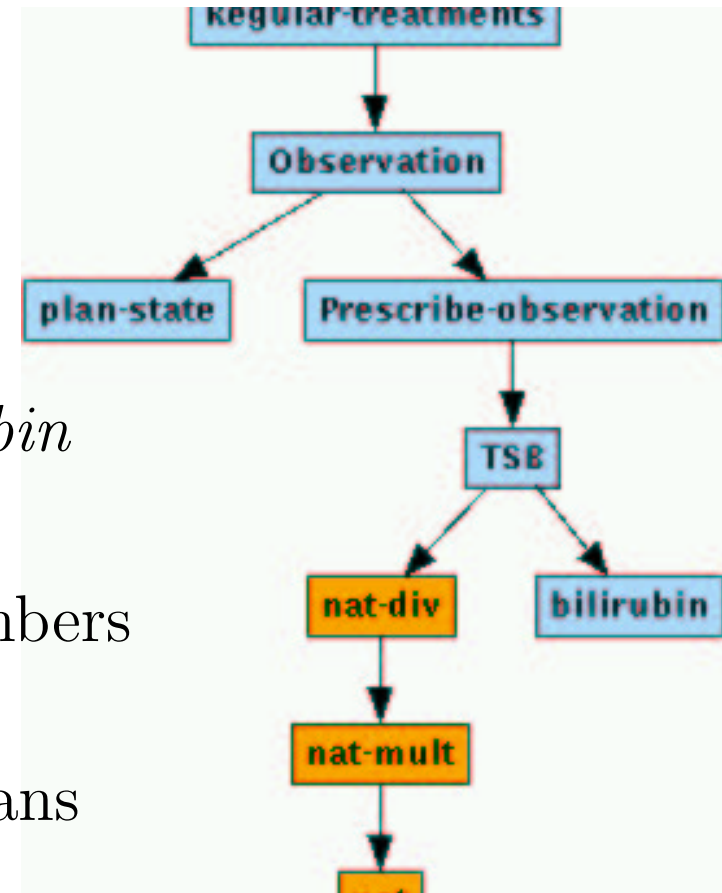
PLAN	Observation (1.4.2.1.5)
PREFERENCES	
INTENTIONS	MAINTAIN INTERMEDIATE STATE bilirubin = observation
CONDITIONS	Filter: (bilirubin observation * [_,_] [0,_] [_,_] NOW) Abort: (bilirubin not observation * [_,_] [0,_] [_,_] NOW)
EFFECTS	
PLAN BODY	Prescribe-Observation
COMMENTS	...

Specification – Overview



⇒ *Development graph*

- Data abstraction
 - ★ Qualitative bilirubin levels ⇒ *bilirubin*
 - ★ Abstraction function ⇒ *TSB*
- Quantitative TSB levels as natural numbers ⇒ *library*
- Hierarchical specification of ASBRU plans ⇒ *modularity*



Specification – Data Abstraction



- Abstract values of bilirubin
⇒ Enumeration type *bilirubin*
- Abstraction function
⇒ Function *get-bilirubin* in *TSB*

Abstraction table

	TSB level	
age	-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



⇒ Specification *Observation*

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

Atomic plan

⇒ Specification *Prescribe-observation*

- Duration is nondeterministic
- Manual activation as nondeterminism

Specification – Properties



- Termination

Observation#(; tsb, time, Observation-state), *System*

□ time'' = time' + 1 *Environment*

⊢ □ get-bilirubin(tsb) = observation → ◇ **last**; *Property*

- Intention

Observation#(; tsb, time, Observation-state), *System*

Observation-state = considered *Environment*

⊢ □ ¬ **last** ∧ Observation-state = activate *Property*

→ get-bilirubin(tsb) = observation;

⇒ 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

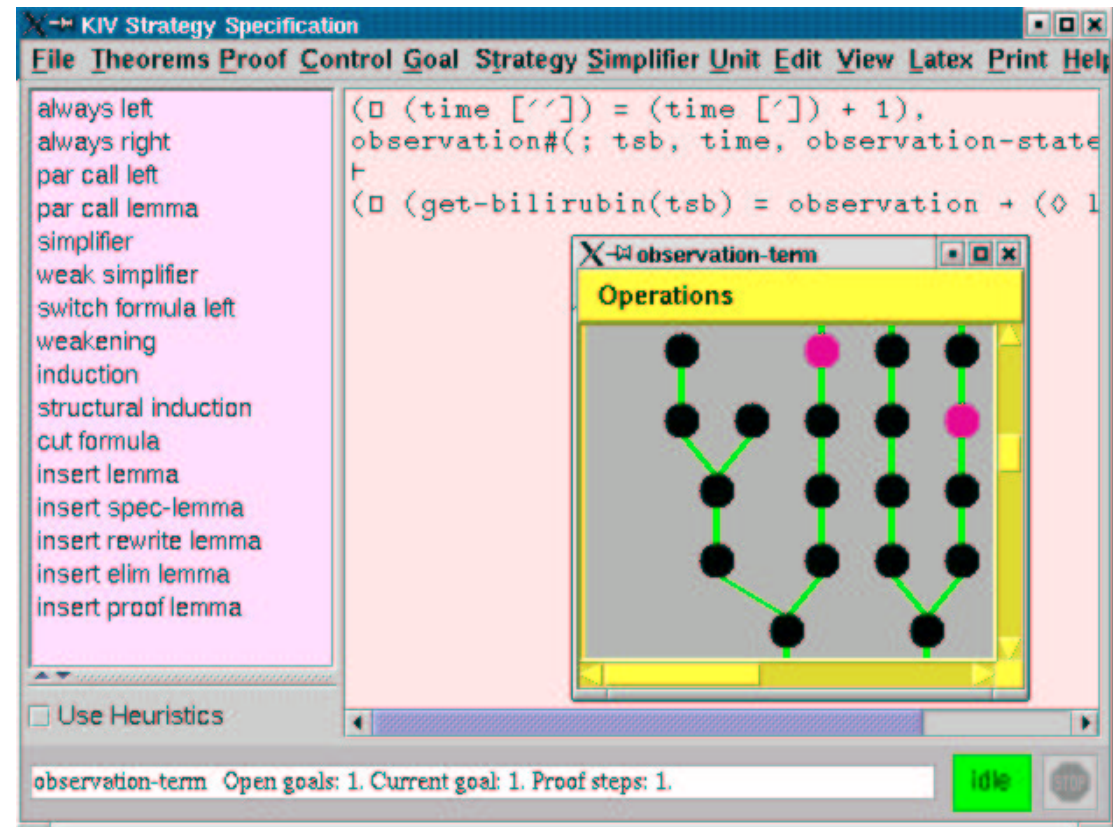


First step

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#(...),

\square Env \vdash \diamond **last**

skip; var ... in await ... ,

\square Env \vdash **last**, \circ \diamond **last**

var ... in await ... ,

\square Env, PL \vdash \diamond **last**

Verification – Induction



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

1. Generalisation

\Rightarrow *weakening*

2. Induction

\Rightarrow *trace induction*

3. Execution of next step

\Rightarrow ...

4. \Rightarrow *apply trace induction*

loc **await** ... ,

□ Env, PL \vdash \diamond **last**

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

□ Env, IndHyp \vdash \diamond **last**

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

□ Env, IndHyp, PL \vdash \diamond **last**

Verification – Modular Proofs



await ...; ...;

break Prescribe-observation#(...); ... **if** ...,

... $\vdash \diamond$ **last**

1. Execute plan until call to sub plan is reached

break Prescribe-observation#(...); ... **if** ..., ... $\vdash \diamond$ **last**

2. Use lemma for sub plan

\Rightarrow *par call lemma*

3. Prove assumption of lemma

break $\neg \square \textit{time}'' = \textit{time}' + 1; \dots$ **if** ..., ... $\vdash \diamond$ **last**

4. Use guarantee of lemma

break \diamond **last**; ... **if** ..., ... $\vdash \diamond$ **last**

Verification – Automation



Heuristics

Heuristics apply rules automatically

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

⇒ *Configure Heuristics*

Degree of Automation

- Work in progress
- Proofs fully automatic except
 - ★ generalisation and
 - ★ 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