

Analysing Z Specifications with HOL-Z

Achim Brucker

SAP Research,
Vincenz-Priessnitz-Str 1
76131 Karlsruhe, Germany
achim.brucker@sap.com

Burkhardt Wolff

Universität des Saarlandes
66041 Saarbrücken, Germany
wolff@wjp.cs.uni-sb.de

Abstract

The increasing complexity of today's software systems makes modeling an important phase during the software development process, both, on the level of requirement analysis and the system design. The ISO-standardized specification language Z can be used for a formal underpinning of these activities. In particular, the Z Method allows for relating Requirements and System Designs via formal refinement notions. In this tutorial we present the interactive theorem prover environment HOL-Z (built as plug-in of Isabelle/HOL) that supports formal reasoning over Z specifications and formal proof on refinements. The system achieved meanwhile a reasonable degree of automation such that several substantial case studies (CVS Server, DARMA funded by Hitachi) had been realized, involving both refinement as well as temporal reasoning.

My Credo's and My Background

- Thesis: THERE IS NO **SINGLE** FORMAL METHOD
- Thesis: FORMAL METHODS MUST BE **INTEGRATED INTO** A (COMPANY-SPECIFIC) SE - **WORKFLOW**
- Thesis: **TOOL-CHAINS** MUST **FOLLOW** METHOD AND WORKFLOW/PRAGMATICS, I.E. THE **METHODOLOGY**.

My Credo's and My Background

- I am a Formal Methods Engineer.

I designed Tool-Chains for:

- process-oriented refinement ("top-down", => HOL-CSP)
- data-oriented refinement ("top-down", => HOL-Z)
- object-oriented refinement ("top-down, MDE", =>HOL-OCL)
- test-oriented ("reverse-engineering", => HOL-TestGen)
- code-verification ("bottom-up", =>HOL-Boogie/C)

according the needs of my "clients"

Outline of HOL-Z Tutorial

- Motivation and Introduction
- Foundations: Z, HOL and Z-Semantics in HOL
- The HOL-Z System
- Advanced Modelling Scenarios
- Theorem Proving in HOL-Z
- Case Studies

Motivation and Introduction

Motivation and Introduction

Why Z?

Motivation and Introduction

Why Z ?

- Z is:
 - a fairly old, but a **mathematically well-defined** FM
 - **ISO standardized** (ISO/IEC 13568:2002, Intern. Standard.)
 - inofficial publication standard for FM papers
 - has nice **text books** (Spivey's "Z Referece Manual", Woodcocks & Davies "Using Z", ...)
 - ... but few **proof-environments**
(CadiZ (experimental), Z/EVES (outdated), ProofPower (HOL4 - based), **HOL-Z** (Isabelle/HOL - based))

Motivation and Introduction

Why Z ?

- what can you do with Z:
 - **top-down refinement** development method (forward-simulation, backward-simulation)
 - generate code, animators (ZAP, ...)
 - it can be used for test-case generation, too.

Motivation and Introduction

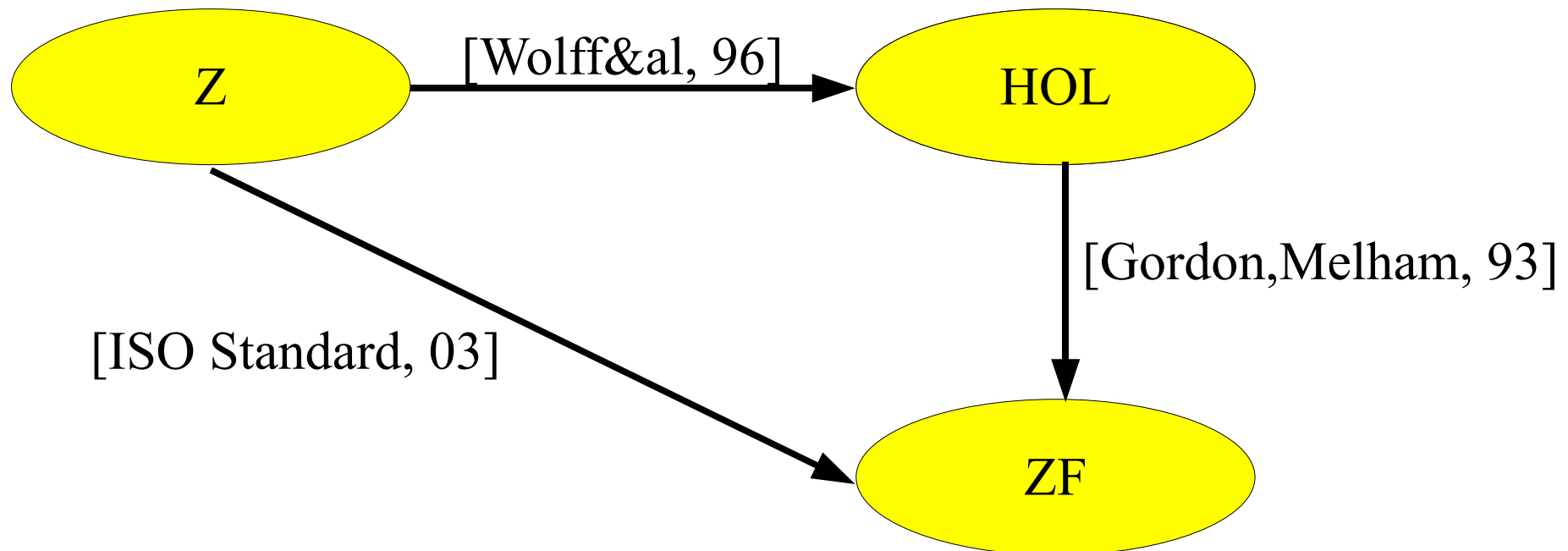
Why Z *in* HOL?

- Z Semantics via Embedding in Higher-Order Logic (HOL)
 - Advantage I: Greatly Simplifies Semantics!
 - Advantage II: Gives Basis for TOOL-SUPPORT within HOL provers (Isabelle, HOL4, ...)

Motivation and Introduction

Why Z in HOL?

- Z Semantics via Embedding
Object Language and a Meta-Language.



Foundations: Z, HOL and Z in HOL

- The Language Z:
 - Z & Types
 - a Guided Tour through the Syntax
- HOL & Embeddings
 - Semantics for Z expressions & predicates in HOL
 - Semantics for Z schema expressions

Foundations: Z, HOL and Z in HOL

Syntax

- Z is:
 - an implicitly simply typed language

$$E :: \tau$$

where the types were given by:

$$\tau ::= \mathbb{Z}$$
$$| \tau \times \cdots \times \tau$$
$$| \langle \text{tag}_1 \rightsquigarrow \tau, \dots, \text{tag}_n \rightsquigarrow \tau \rangle$$
$$| \mathbb{P}(\tau)$$

Foundations: Z, HOL and Z in HOL

Semantics

- HOL is:
 - a simply typed language based on λ -terms:

$$E ::= C \mid V \mid \lambda x. E \mid E E$$

$$E :: \tau$$

where $\tau ::= \alpha \mid \tau \rightarrow \tau \mid \chi(\tau, \dots, \tau)$

Foundations: Z, HOL and Z in HOL

Semantics

- HOL has:
 - declaration principles:

arities $\chi ::$ “kind-declaration”

int, bool, α set, α list, ...

consts $c ::$ “ τ ”

True, False :: “bool”

$_ = _ ::$ “ $\alpha \rightarrow \alpha \rightarrow$ bool”

$_ \vee _, _ \wedge _ ::$ “bool \rightarrow bool \rightarrow bool”

0 :: “nat”,

insert :: “ $\alpha \rightarrow \alpha$ set \rightarrow set”, ...

Foundations: Z, HOL and Z in HOL

Semantics

- HOL has:

- axioms (for a core of 9 axioms only):

axiom <name> : “E”

axiom refl : “ $x = x$ ”

sym : “ $s = t \implies t = s$ ”

mp : “ $P \implies P \rightarrow Q \implies Q$ ”

subst: “ $\llbracket s=t; P s \rrbracket \implies P t$ ”

...

where \implies is the meta-implication. We write

$\llbracket A_1; \dots; A_n \rrbracket \implies B$ for $A_1 \implies \dots \implies A_n \implies B$.

Foundations: Z, HOL and Z in HOL

Semantics

- HOL has:
 - conservative extension schemes like “constant definition”:

defs <name> : “c \equiv E”

where E closed and constant c “fresh”

defs All_def : “ $\forall \equiv \lambda P. (P = \lambda x. \text{True})$ ”

(we write $\forall x. P x$ for $\forall(\lambda x. P x)$...)

Foundations: Z, HOL and Z in HOL

Semantics

- HOL has:
 - conservative extension schemes like “type definition”:

typedef $\alpha \chi = \{x::\tau \mid E\}$

where E closed and χ “fresh”.

Foundations: Z, HOL and Z in HOL

Semantics

- HOL has:
 - conservative extension schemes like “type definition”:

It abbreviates:

arities $\alpha \chi$ “...”

consts $\text{Abs}_\chi :: \tau \text{ set} \rightarrow \alpha \chi$

$\text{Rep}_\chi :: \alpha \chi \rightarrow \tau \text{ set}$

axioms $\text{Abs}_\chi\text{-inverse} : \text{“Abs}_\chi (\text{Rep}_\chi x) = x\text{”}$

$\text{Rep}_\chi\text{-inverse} : \text{“}\exists x \Rightarrow$

$\text{Rep}_\chi (\text{Abs}_\chi x) = x\text{”}$

Foundations: Z, HOL and Z in HOL

Semantics

- HOL vs. Z : a first summary
 - Z: a rich language with a particular system modeling methodology (as we will see!)
 - HOL : minimalistic (small language, few powerful principles), emphasis on clean foundations.

HOL is the MACH of the Logics !!!

From Foundations to Pragmatics

Semantics

- HOL is based on conservative extension schemes. This
 - **guarantees consistency** provided that "core HOL" is consistent
 - is still **expressive enough**:
entire HOL-library comprising theories on sets, orderings, numbers, cartesian products, type sums, recursion, data-types is **derived from conservative definitions** and the core axioms !!!

Foundations: Z, HOL and Z in HOL

Syntax

- (typed) Expressions E in Z are:
 - arithmetic: $1, 2, 3, a + b, a / b, \dots$
 - pairs: (a_1, \dots, a_n) , pattern-abstractions: $\lambda(a_1, \dots, a_n).E$
 - records: $\langle \text{tag}_1 \rightsquigarrow E, \dots, \text{tag}_n \rightsquigarrow E \rangle$

Foundations: Z, HOL and Z in HOL

Semantics

- (typed) Expressions E in Z are:
 - arithmetic: $1, 2, 3, a + b, a / b, \dots$
semantics: Library Theory Arith with type
 $\text{type int} = \mathbb{Z}$
 - pairs: (a_1, \dots, a_n) , pattern-abstractions: $\lambda(a_1, \dots, a_n).E$
semantics: Library Theory Pair with type $_ \times _$
 - records: $\langle \text{tag}_1 \rightsquigarrow \tau, \dots, \text{tag}_n \rightsquigarrow \tau \rangle$
semantics: Library Theory Pair with type
plus some own pre-computation/
reordering in HOL-Z

Foundations: Z, HOL and Z in HOL

Syntax

- (typed) Expressions E in Z are:
 - set constants : \mathbb{N}, Z ($:: P(Z)$!!!)
 - set constructors : $a \in B, \{a \in B \mid P(a) \bullet f(a)\},$

Foundations: Z, HOL and Z in HOL

Semantics

- (typed) Expressions E in Z are:

– **set constants** : \mathbb{N}, \mathbb{Z} ($:: P(\mathbb{Z})$!!!)

constdefs $\mathbb{N} \equiv \{a :: \text{int} \mid 0 \leq a\}, \dots$

$\mathbb{Z} \equiv \{a :: \text{int} \mid \text{True}\}, \dots$

(HOL comprehension!)

– **set constructors** : $a \in B, \{a \in B \mid P(a) \bullet f(a)\},$

constdefs “ $f' S \equiv \{a :: \text{int} \mid \exists y. a = f y\}$ ”

“ $\{a \in B \mid P(a) \bullet f(a)\} \equiv$

$f' \{x \mid a \in B \wedge P(a)\}$ ”

Foundations: Z, HOL and Z in HOL

Syntax

- (typed) Expressions E in Z are:
 - set predicates: $A \subseteq B$
 - set operators: $A \cup B, A \cap B, A \times B, R \circ S,$
 - set operators: $F(A), P(A)$ ($P :: P(P(A) \times PP(A)) !!!$)

Foundations: Z, HOL and Z in HOL

Syntax

- (typed) Expressions E in Z are:
 - **set predicates:** $A \subseteq B$
constdefs $A \subseteq B \equiv \forall a. a \in A \rightarrow a \in B$
 - **set operators:** $A \cup B, A \cap B, A \times B, R \circ S, R \oplus S$
constdefs $A \cup B \equiv \{a \mid a \in A \vee a \in B\}$
 - **set operators:** $\mathcal{P}(A)$ ($\mathcal{P} :: \mathcal{P}(\mathcal{P}(A) \times \mathcal{P}\mathcal{P}(A)) \implies$), $\mathbb{F}(A)$
constdefs $\mathcal{P}(A) \equiv \{B \mid B \subseteq A\}$
 $\mathbb{F}(A) \equiv \{B \mid B \subseteq A \wedge \text{finite } B\}$

Foundations: Z, HOL and Z in HOL

Semantics

- (typed) Expressions E in Z are:

- “function” constructors:

$A \leftrightarrow B$: relation from A to B

$A \rightarrow B$: partial function from A to B

$A \twoheadrightarrow B$: total function from A to B

$A \xrightarrow{\sim} B$: bijective partial function from A to B

$A \twoheadrightarrow B$: total finite bijection from A to B

$A \twoheadrightarrow B$: partial finite surjection from A to B

...

Foundations: Z, HOL and Z in HOL

Syntax

- (typed) Expressions E in Z are:
 - "function" constructors:

$$A \leftrightarrow B \equiv \{r \mid r \subseteq A \times B\}$$

$$A \rightarrow B \equiv \{r \in A \leftrightarrow B \mid \forall a \in \text{dom } r. x, y \in B. \\ (a, x) \in r \wedge (a, y) \in r \rightarrow x = y\}$$

$$A \rightarrow B \equiv \dots$$

$$A \rightarrow B \equiv \dots$$

$$A \rightarrow B \equiv \dots$$

$$A \rightarrow B \equiv \dots$$

...

Foundations: Z, HOL and Z in HOL

Syntax

- Predicates P in Z are:
 - $E = E, x \in E, E \subseteq E, \dots$
 - $\forall x \in E \bullet E, \exists x \in E \bullet E, \dots$
 - $E \Leftrightarrow E, E \vee E, E \wedge E, E \rightarrow E, \neg E, \dots$

Foundations: Z, HOL and Z in HOL

Syntax

- Predicates P in Z are:

as in HOL

Foundations: Z, HOL and Z in HOL

Syntax

- Toplevel-Constructs

- abstract types

[book, date, user]

- axiomatic definitions (1)

$$\frac{_ \geq _ : \mathbb{P} (\text{date} \times \text{date})}{\text{total_ordering } (_ \geq _)}$$

- axiomatic definitions (2)

$$x \equiv E$$

Foundations: Z, HOL and Z in HOL

Semantics

- Toplevel-Constructs

- abstract types

arities book :: “...” ...

- axiomatic definitions (1)

consts $_ \geq _$: (date \times date) set
axiom order_axdef :

“ $_ \geq _ \in \mathbb{P}(\text{date} \times \text{date}) \wedge$
total_ordering ($_ \geq _$)”

- axiomatic definitions (2)

$x \equiv E$

Foundations: Z, HOL and Z in HOL

Syntax

- Toplevel-Constructs
 - schema declarations:

A	
$d_1 : T_1$	
...	-- declaration part
$d_n : T_n$	
<hr/>	
$P(d_1, \dots, d_n)$	-- predicative part

where A is now considered to special lexical class called **schema names**.

Foundations: Z, HOL and Z in HOL

Semantics

- Toplevel-Constructs
 - schema declarations:

A \equiv

$d_1 : T_1$

...

$d_n : T_n$

-- declaration part

$P(d_1, \dots, d_n)$

-- predicative part

Foundations: Z, HOL and Z in HOL

Semantics

- Toplevel-Constructs
 - schema declarations:

ISO-STANDARD:

$$A \equiv \{ d_1 : T_1; \dots; d_n : T_n \mid P(d_1, \dots, d_n) \bullet \langle d_1 \rightsquigarrow d_1, \dots, d_n \rightsquigarrow d_1 \rangle \}$$

From Foundations to Pragmatics

Semantics

- Toplevel-Constructs
 - schema declarations:

EQUIVALENTLY :

SCHEMAS AS FUNCTIONS:

$$A \equiv \lambda(a_1, \dots, a_n) \cdot a_1 \in T_1 \wedge \dots \wedge a_n \in T_n \\ \wedge P(a_1, \dots, a_n)$$

From Foundations to Pragmatics

Semantics

- Toplevel-Constructs
 - schema declarations:

OUR VERSION IN HOL-Z
(robust against alpha conversion!)

$$A \equiv \text{SB } "a_1" \rightsquigarrow a_1, \dots, "a_n" \rightsquigarrow a_n \cdot \quad a_1 \in T_1 \wedge \dots \wedge a_n \in T_n \\ \wedge P(a_1, \dots, a_n)$$

Foundations: Z, HOL and Z in HOL

Syntax

- Toplevel-Constructs: schemas with imports:

A
$x_1 : S_1$ $x_2 : S_2$
$P(x_1, x_2)$

B
$A; A';$ $x_2 : T$
$Q(x_1, x_2, x'_1, x'_2)$

C
$B;$ $z : \text{seq}(A)$
$R(x_1, x_2, x'_1, x'_2, z)$

Foundations: Z, HOL and Z in HOL

Syntax

- Toplevel-Constructs: schemas with imports:

$$A \equiv \lambda(x_1, x_2) \cdot \\ x_1 \in S_1 \wedge x_2 \in S_2 \\ \wedge P(x_1, x_2)$$

$$\frac{B}{\begin{array}{l} A; A'; \\ x_2 : T \end{array}} \\ \frac{}{Q(x_1, x_2, x'_1, x'_2)}$$

$$\frac{C}{\begin{array}{l} B; \\ z : \text{seq}(A) \end{array}} \\ \frac{}{R(x_1, x_2, x'_1, x'_2, z)}$$

Foundations: Z, HOL and Z in HOL

Syntax

- Toplevel-Constructs: schemas with imports:

$$A \equiv \lambda(x_1, x_2).$$

$$x_1 \in S_1 \wedge x_2 \in S_2 \\ \wedge P(x_1, x_2)$$

$$B \equiv \lambda(x_1, x_2, x'_1, x'_2).$$

$$A(x_1, x_2) \wedge A(x'_1, x'_2) \wedge \\ x_2 \in T \wedge \\ Q(x_1, x_2, x'_1, x'_2)$$

$$\frac{\text{C}}{\text{B}; \\ z : \text{seq}(A)} \\ \hline \text{R}(x_1, x_2, x'_1, x'_2, z)$$

Foundations: Z, HOL and Z in HOL

Syntax

- Toplevel-Constructs: schemas with imports:

$$A \equiv \lambda(x_1, x_2) \cdot$$

$$x_1 \in S_1 \wedge x_2 \in S_2 \\ \wedge P(x_1, x_2)$$

$$B \equiv \lambda(x_1, x_2, x'_1, x'_2) \cdot$$

$$A(x_1, x_2) \wedge A(x'_1, x'_2) \wedge \\ x_2 \in T \wedge \\ Q(x_1, x_2, x'_1, x'_2)$$

$$C \equiv \lambda(x_1, x_2, x'_1, x'_2, z) \cdot$$

$$B(x_1, x_2, x'_1, x'_2) \wedge \\ x_2 \in \text{seq}(\text{asSet } A) \wedge \\ R(x_1, x_2, x'_1, x'_2, z)$$

Foundations: Z, HOL and Z in HOL

Syntax

- Toplevel-Constructs: schemas with imports:

$$\begin{array}{l}
 A \equiv \text{SB } \text{"x}_1\text{"} \rightsquigarrow x_1, \text{"x}_2\text{"} \rightsquigarrow x_2. \\
 x_1 \in S_1 \wedge x_2 \in S_2 \\
 \wedge P(x_1, x_2)
 \end{array}
 \quad
 \begin{array}{l}
 B \equiv \text{SB } \text{"x}_1\text{"} \rightsquigarrow x_1, \text{"x}_2\text{"} \rightsquigarrow x_2 \\
 \text{"x}'_1\text{"} \rightsquigarrow x'_1, \text{"x}'_2\text{"} \rightsquigarrow x'_2. \\
 A(x_1, x_2) \wedge A(x'_1, x'_2) \wedge \\
 x_2 \in T \wedge \\
 Q(x_1, x_2, x'_1, x'_2)
 \end{array}$$

$$\begin{array}{l}
 C \equiv \text{SB } \text{"x}_1\text{"} \rightsquigarrow x_1, \text{"x}_2\text{"} \rightsquigarrow x_2 \text{"x}'_1\text{"} \rightsquigarrow x'_1, \text{"x}'_2\text{"} \rightsquigarrow x'_2 \text{"z"} \rightsquigarrow z. \\
 B(x_1, x_2, x'_1, x'_2) \wedge \\
 x_2 \in \text{seq}(\text{asSet } A) \wedge \\
 R(x_1, x_2, x'_1, x'_2, z)
 \end{array}$$

Foundations: Z, HOL and Z in HOL

Syntax

- Toplevel-Constructs: **Schema Calculus**
 - schema decoration: S'
 ΔS
 $\exists S$
 - schema expressions: $A \wedge B$
 $S \Leftrightarrow S, S \vee S,$
 $S \rightarrow S, \neg S, \dots$
 - schema quantification $\forall S \bullet S, \exists S \bullet S, \dots$

Foundations: Z, HOL and Z in HOL

Semantics

- Toplevel-Constructs: **Schema Calculus**
 - schema decoration: S' renaming signature (as bef.)
 $\Delta S \equiv S \wedge S'$
 $\Xi S \equiv \Delta S \wedge (x_1 = x'_1 \wedge \dots \wedge x_n = x'_n)$
 - schema expressions: $A \wedge B \equiv \lambda(x_1 \dots x_n). A(x_{i1}, \dots, x_{im}) \wedge B(x_{j1}, \dots, x_{jm})$
 $S \Leftrightarrow S, S \vee S, \dots$
 $S \rightarrow S, \neg S, \dots \dots$
 - schema quantification $\forall S \bullet S, \exists S \bullet S, \dots$

Foundations: Z, HOL and Z in HOL

Syntax and Semantics

- Summary:
 - HOL is a good (simple) Meta-Language for Z, even for the Schema-Calculus
 - schemas are “formulas with a signature” (binding)
 - schema join:
 - **union** of signatures
 - **conjunction** of predicates
 - HOL-Z: schemas were represented as characteristic functions - the structure is preserved (no unfolding, flattening, etc, ...)

The HOL-Z System

- Based on an advanced, interactive proof-environment: Isabelle/HOL
- Conceived as Plugin into the Isabelle/ISAR architecture
- Provides Parser, conservative semantic theories, own proof procedures
- A refinement package to support top-down development

The HOL-Z System

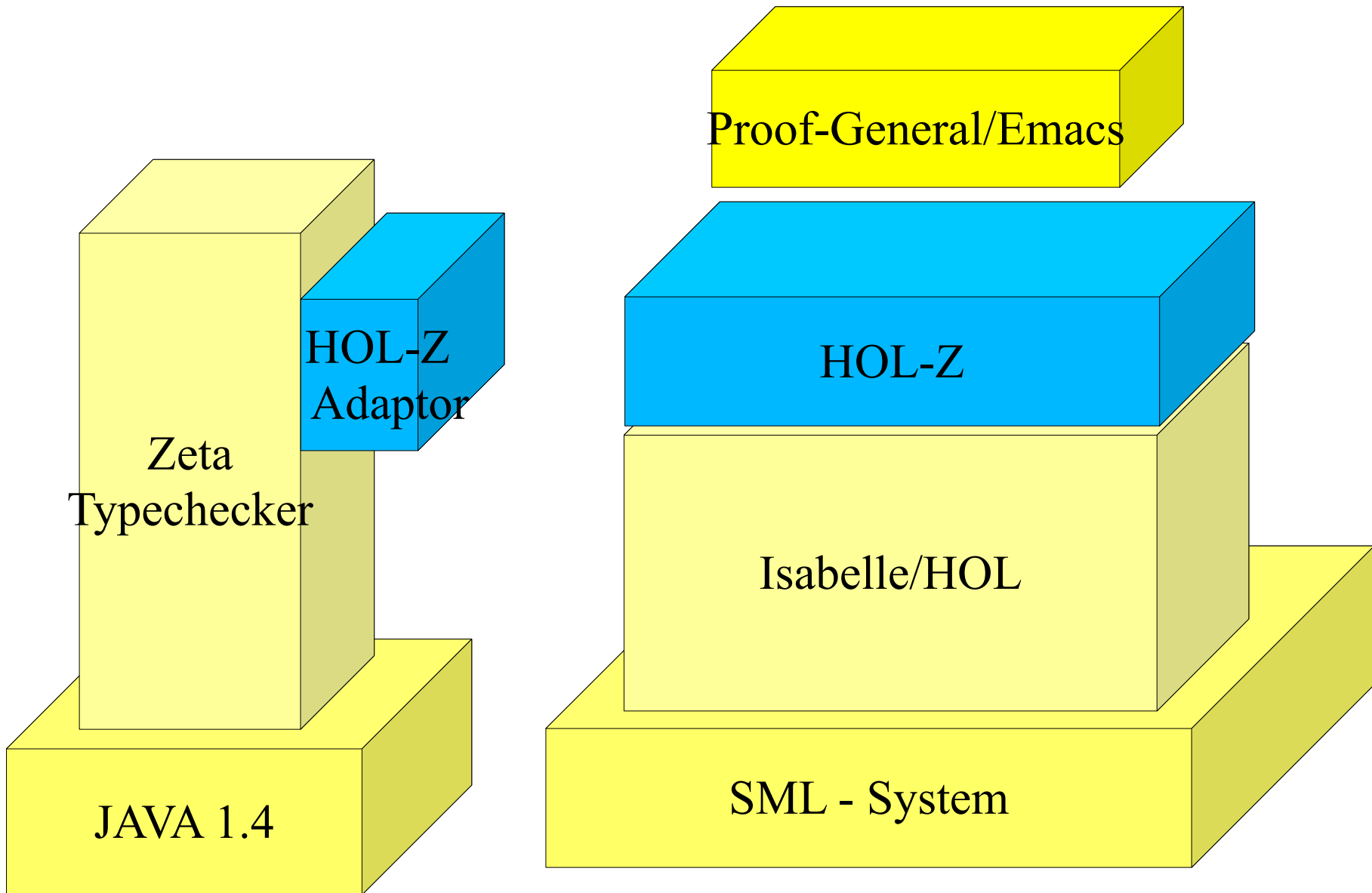
- The HOL-Z System Architecture
- The Workflow

Underlying Assumption:

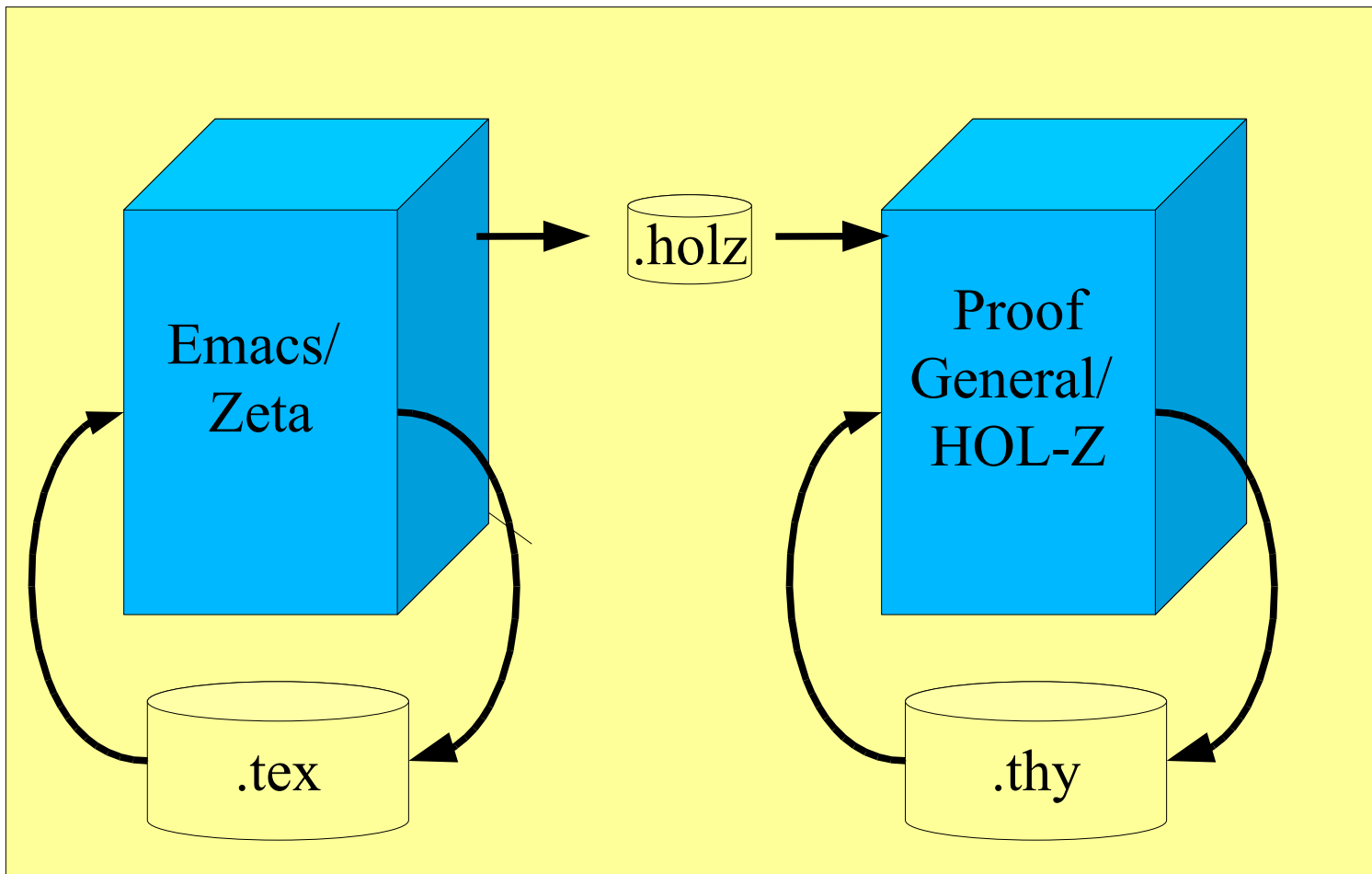
Designer and Proof-Engineer not necessarily the same person !

The HOL-Z System

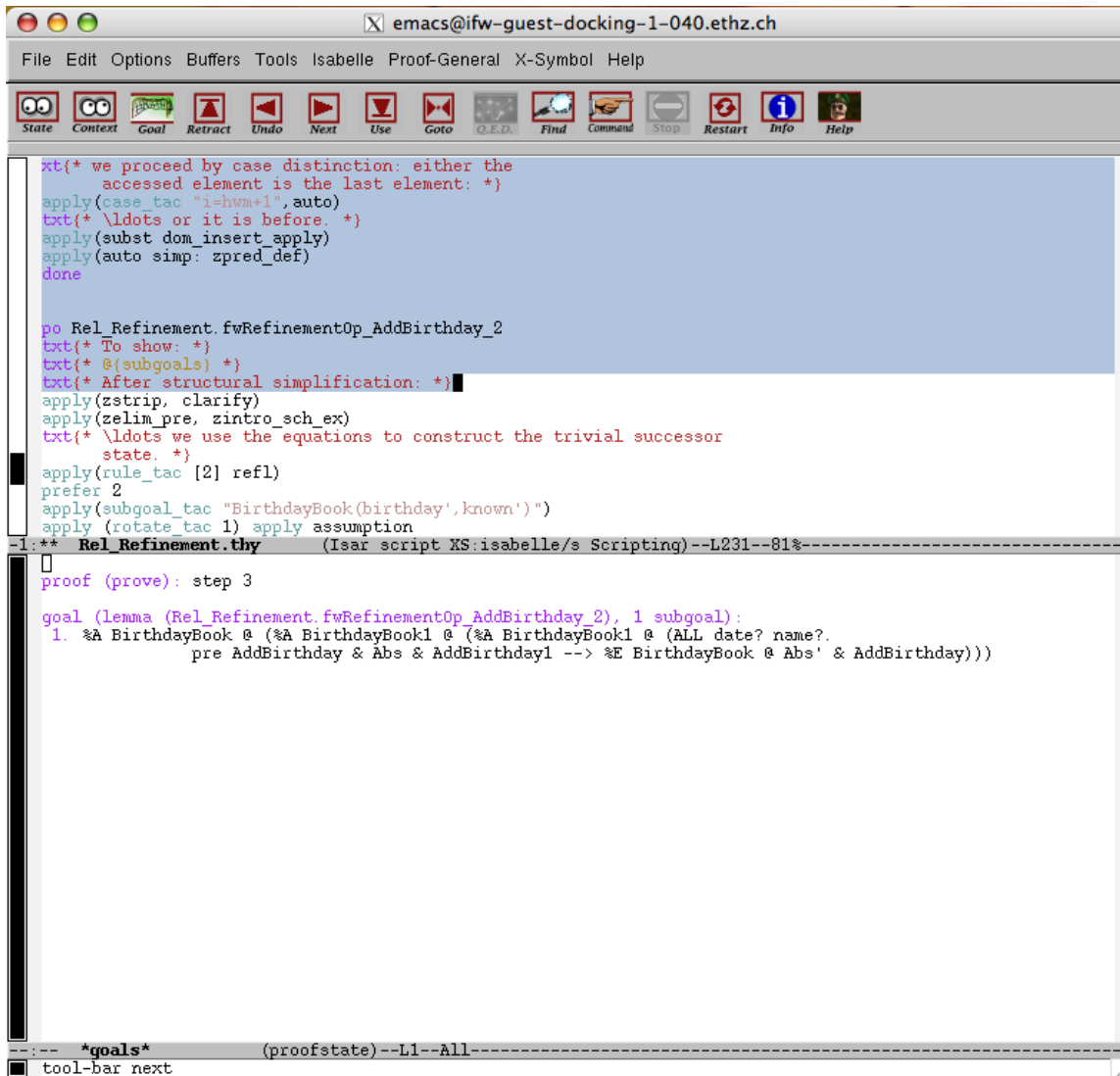
Component View



The HOL-Z System Workflow View



The HOL-Z System ProofGeneral



The screenshot shows the Emacs editor window titled 'emacs@ifw-guest-docking-1-040.ethz.ch'. The menu bar includes 'File Edit Options Buffers Tools Isabelle Proof-General X-Symbol Help'. The toolbar contains icons for 'State', 'Context', 'Goal', 'Retract', 'Undo', 'Next', 'Use', 'Goto', 'Q.E.D.', 'Find', 'Command', 'Stop', 'Restart', 'Info', and 'Help'. The main text area contains the following code:

```
xt{* we proceed by case distinction: either the
   accessed element is the last element: *}
apply(case_tac "i-hvm+1", auto)
txt{* \ldots or it is before. *}
apply(subst dom_insert_apply)
apply(auto simp: zpred_def)
done

po Rel_Refinement.fwRefinementOp_AddBirthday_2
txt{* To show: *}
txt{* @ {subgoals} *}
txt{* After structural simplification: *}
apply(zstrip, clarify)
apply(zelim_pre, zintro_sch_ex)
txt{* \ldots we use the equations to construct the trivial successor
      state. *}
apply(rule_tac [2] refl)
prefer 2
apply(subgoal_tac "BirthdayBook(birthday', known')")
apply (rotate_tac 1) apply assumption
-1:** Rel_Refinement.thy (Isar script XS:isabelle/s Scripting)--L231--81%-----
proof (prove): step 3
goal (lemma (Rel_Refinement.fwRefinementOp_AddBirthday 2), 1 subgoal):
1. %A BirthdayBook @ (%A BirthdayBook1 @ (%A BirthdayBook1 @ (ALL date? name?.
   pre AddBirthday & Abs & AddBirthday1 --> %E BirthdayBook @ Abs' & AddBirthday)))
-:-- *goals* (proofstate)--L1--All
tool-bar next
```



checked area
(non-editible)



unchecked area
(editable)



system reaction
(current proof state,
current theory state,
errors ...)

The HOL-Z System

Zeta

- Zeta is conceived as a type-checker on Z documents
- Z documents were written by LaTeX Markups
- Proof documents were written:
 - by e-mail format for Z (in future obsolete)
 - by LaTeX Markups
 - and ISAR commands (for proof tactics)

The HOL-Z System

Zeta

- Zeta is conceived as a type-checker on Z documents
- Z documents were written by LaTeX Markups
- Proof documents were written:
 - by e-mail format for Z (in future obsolete)
 - by LaTeX Markups
 - and ISAR commands (for proof tactics)

The HOL-Z System

Z LaTeX : Logic

• Math	HOL-Z	ZETA	
\neg	\sim , not, <code>\<Inot></code>	<code>\Inot</code>	unary
pre	PRE	<code>\pre</code>	unary
\wedge	<code>&</code> , <code>\<land></code> , \wedge	<code>\land</code>	left
\vee	<code> </code> , <code>\<lor></code> , \vee	<code>\lor</code>	left
\rightarrow	<code>--></code> , <code>\<implies></code>	<code>\implies</code>	right
\Leftrightarrow	<code>=</code>	<code>\iff</code>	left
\forall	<code>!</code> , <code>\<forall></code>	<code>\forall</code>	
\exists	<code>?</code> , <code>\<exists></code>	<code>\exists</code>	
\uparrow	project	<code>\project</code>	left
\backslash	hide	<code>\hide</code>	left
$;$	not supported	<code>\semi</code>	left
$>>$	not supported	<code>\pipe</code>	left
if	if	<code>\IF</code>	
then	then	<code>\THEN</code>	
else	else	<code>\ELSE</code>	
let	let	<code>\LET</code>	

The HOL-Z System

Z LaTeX : Sets

• Math	HOL-Z	ZETA	
\mathbb{P}	Pow, \<power>	\power	pregen
\mathbb{F}	Fin, \<finset>	\finset	pregen
\emptyset	\}	\emptyset	word
#	card	\#	word
\cap	Int, \<cap>	\cap	inop 4
\cup	Un, \<cup>	\cup	inop 4
-	-	\setminus	inop 3
\subseteq	<=, \<subsepeq>	\subsepeq	inrel
\subset	<	\subset	inrel
=	=	= =	inrel
\neq	\neq	\neq	inrel
\in	:, \<in>	\in	inrel
\notin	\notin	\notin	inrel

The HOL-Z System

Z LaTeX : Relations and Functions

• Math	HOL-Z	ZETA	
\mapsto	<code> -></code> , <code>\<mapsto></code>	<code>\mapsto</code>	inop 1
\times	*	<code>\cross</code>	inop 1
dom	dom	<code>\dom</code>	word
ran	ran	<code>\ran</code>	word
\circ	o	<code>\circ</code>	inop 4
;	<code>\<semi></code>	<code>\comp</code>	inop 5
\oplus	(+), <code>\<oplus></code>	<code>\oplus</code>	inop 5
\sim	not supported	<code>\inv</code>	postop
\triangleleft	<code>< </code> , <code>\<dres></code>	<code>\dres</code>	inop 6
\triangleright	<code> ></code> , <code>\<rres></code>	<code>\rres</code>	inop 6
\triangleleft	<code><- </code> , <code>\<ndres></code>	<code>\ndres</code>	inop 6
\triangleright	<code> -></code> , <code>\<nrres></code>	<code>\nrres</code>	inop 6
+	not supported	<code>\plus</code>	postop
*	not supported	<code>\star</code>	postop

The HOL-Z System

Z LaTeX : Relations and Functions

• Math	HOL-Z	ZETA	
$f(x)$	<code>f%^x,f\<rappl>x\<rapplr></code>	<code>f(x)</code>	(* relational application *)
*	<code>\<star></code>	<code>\star</code>	postop
\leftrightarrow	<code><->, \<rel></code>	<code>\rel</code>	ingen
\rightarrow	<code>- ->, \<pfun></code>	<code>\pfun</code>	ingen
\rightarrow	<code>--->, \<fun></code>	<code>\fun</code>	ingen
\rightarrow	<code>>- ->, \<pinj></code>	<code>\pinj</code>	ingen
\rightarrow	<code>>--->, \<inj></code>	<code>\inj</code>	ingen
\rightarrow	<code>- ->>, \<psurf></code>	<code>\psurf</code>	ingen
\rightarrow	<code>-->>, \<surj></code>	<code>\surj</code>	ingen
\rightarrow	<code>>--->, \<bij></code>	<code>\bij</code>	ingen
\rightarrow	<code>- -> \<ffun></code>	<code>\ffun</code>	ingen
\rightarrow	<code>>- -> , \<finj></code>	<code>\finj</code>	ingen
id	id	<code>\id</code>	word
(<code>\<limg></code>	<code>\limg</code>	-
)	<code>\<rimg></code>	<code>\rimg</code>	-

The HOL-Z System

Z LaTeX : Integers

• Math	HOL-Z	ZETA	
\mathbb{Z}	<code>%Z,\<num></code>	<code>\num</code>	word
\mathbb{N}	<code>%N,\<nat></code>	<code>\nat</code>	word
\dots	<code>.., \<upto></code>	<code>\upto</code>	inop 2
$+$	<code>+</code>	<code>+</code>	inop 3
$-$	<code>-</code>	<code>-</code>	inop 3
$*$	<code>*</code>	<code>*</code>	inop 4
div	<code>div</code>	<code>\div</code>	inop 4
mod	<code>modl</code>	<code>\mod</code>	inop 4
$<$	<code><</code>	<code><</code>	inrel
\leq	<code><=,\<le></code>	<code>\leq</code>	inrel
$>$	<code>>,</code>	<code>></code>	inrel
\geq	<code>>=,\<geq></code>	<code>\geq</code>	inrel

The HOL-Z System

Z LaTeX : Integers

• Math	HOL-Z	ZETA	
<		\langle	-
>		\rangle	-
seq	seq	\seq	pregen
iseq	iseq	\iseq	pregen
in	seqin	\inseq	inrel
~	seqconcat	\cat	inop 3
prefix	prefix	\prefix	inrel
suffix	suffix	\suffix	inrel
↑	↑	\filter	word
??	↑	\extract	word

The HOL-Z System

Z LaTeX : Toplevel

- Math HOL-Z ZETA

- Schema Definitions

A	constdef <name>: ...	<code>\begin[A]{schema}</code>
$x_1 : S_1$		<code> x_1 : S1;</code>
$x_2 : S_2$		<code> x_2 : S2</code>
$P(x_1, x_2)$		<code> \where</code> <code> P(x_1, x_2)</code>
		<code> \end{axdef}</code>

DEMO

Theorem Proving in HOL-Z

- Introduction Theorem Proving in Isabelle/HOL
- HOL-Z library
- HOL-Z specific proof methods

Theorem Proving in HOL-Z

Intro: Isabelle/HOL

- Hierarchical Proof Documents

```
theory X
imports Y Z
begin
  ...
end
```

Theorem Proving in HOL-Z

Intro: Isabelle/HOL

- Elements in theories are:
 - arities (seen before)
 - consts (seen before)
 - constdefs (seen before)
 - axioms (seen before)
 - toplevel commands
 - declare thm [simp]
 - declare thm [intro!]
 - thm name ...

Theorem Proving in HOL-Z

Intro: Isabelle/HOL

- Elements in theories are:
 - proofs

```
lemma name[modifier]:  
  " E "  
  apply(method )  
  ...  
  proof (method)  
  done
```

Theorem Proving in HOL-Z

Intro: Isabelle/HOL

- Elements in theories are:
 - proofs

```
lemma name[modifier]:  
  “ E “  
  have A : “sublemma” ...  
    ...  
  have Z : “sublemma” ...  
  show ?thesis:  
    apply(...) ... done  
qed
```

Theorem Proving in HOL-Z

Intro: Isabelle/HOL

- Proof Methods:

- unfolding unfold ...
- inserting a theorem
 into assumptions insert
- one-step-rewriting subst
- resolution rule, drule, erule, frule
- simplification simp
- tableau reasoner auto

Theorem Proving in HOL-Z

Intro: Isabelle/HOL

- More can be found in the "Isabelle Book":

LNCS 2283:

T. Nipkow, L. C. Paulson, M. Wenzel:
Isabelle/HOL A Proof Assistant for
Higher-Order Logic,

- ... or the excellent system documentation!

<http://isabelle.in.tum.de/>

Theorem Proving in HOL-Z

- The HOL/ISAR command language is extended by HOL-Z specific commands:

- import of ZeTa-Models:

use_holz “<holz>”

- new modifier:

<thm> [zstrip]

<thm> [zdecl[no]]

<thm> [pred[no]]

- new attributes:

declare <thm>[tc]

Theorem Proving in HOL-Z

- The HOL/ISAR command language is extended by HOL-Z specific commands:
 - new methods
(stripping Z representation before use)
 - tc (infers typing predicates from declaration parts ...)
 - zunfold <thm>
 - zfullunfold <thm>
 - zrule <thm>, zdrule <thm>, zerule<thm>
 - zsubst <thm>

Theorem Proving in HOL-Z

- The HOL/ISAR command language is extended by HOL-Z specific commands:
 - new methods supporting schema calculus:

$$\frac{\bigwedge \langle x \rangle . S(\langle x \rangle)}{\vdash S}$$

zturnstyleI

$$\frac{\begin{array}{c} [S(\langle ?x \rangle)] \\ : \\ \vdash S \quad R \end{array}}{R}$$

zturnstyleE

Theorem Proving in HOL-Z

- The HOL/ISAR command language is extended by HOL-Z specific commands:
 - new methods supporting schema calculus:

$$\frac{\begin{array}{c} [S(\langle x \rangle)] \\ \vdots \\ \bigwedge \langle x \rangle. \quad T(\langle x \rangle \langle y \rangle) \end{array}}{\forall S \bullet T} \quad (*)$$

zschalll

$$\frac{\begin{array}{c} [S(\langle ?x \rangle), T(\langle ?x \rangle \langle y \rangle)] \\ \vdots \\ \forall S \bullet T \quad R \end{array}}{R} \quad (*)$$

zschallE

Theorem Proving in HOL-Z

- The HOL/ISAR command language is extended by HOL-Z specific commands:
 - new methods supporting schema calculus:

$$\begin{array}{c}
 \frac{S(\langle y \rangle, \langle x' \rangle, \langle x! \rangle)}{\text{pre } S} \quad (*) \\
 \text{zprel}
 \end{array}
 \qquad
 \frac{\text{pre } S \quad \wedge \langle x' \rangle \langle x! \rangle. \quad R}{R} \quad (*) \quad [S(\langle y \rangle, \langle x' \rangle, \langle x! \rangle)] \\
 \text{zpreE}$$

Theorem Proving in HOL-Z

- The HOL/ISAR command language is extended by HOL-Z specific commands:
 - new methods supporting schema calculus:

$$\frac{S(\langle x \rangle) \quad T(\langle x \rangle \langle y \rangle)}{\exists S \bullet T} \qquad \frac{\exists S \bullet T \quad \wedge \langle x \rangle. R}{R} \quad [S(\langle x \rangle); T(\langle x \rangle \langle y \rangle)]$$

zschem1

zschemE

Theorem Proving in HOL-Z

- Notation/Provisos

of the previous schema rule schemes:

- $\langle x \rangle, \langle x' \rangle, \langle x! \rangle, \langle ?x \rangle$ denote vectors of variables (primed variables, successor state variables, output variables, meta variables) x_1, \dots, x_n
- $\langle x \rangle \langle y \rangle$ denotes the concatenation of these vectors
- $\langle \underline{x} \rangle$ denotes a permutation of a vector
- $(^*)$ stands for the proviso: $\langle y \rangle$ is a vector of free variables (not occurring in the hypothesis) corresponding to the schema signature of the conclusion (in the introduction rules or the first premise (in the elimination rules)).

Advanced Modelling Scenarios

- Analysis (consistency, implementability)
- Refinement
- Modeling Temporal Properties

Advanced Modelling Scenarios

Refinement

- **Concept (Top Down Development):**
Refine each operation op_{abs} of a transition system $sys_{abs} = (\sigma_{abs}, init_{abs}, op_{abs})$ to a more concrete system $sys_{conc} = (\sigma_{conc}, init_{conc}, op_{conc})$.
Chain the refinements:

$$sys_1 \rightsquigarrow sys_2 \rightsquigarrow \dots \rightsquigarrow sys_n$$

to a version sys_n amenable to a code generator.

Advanced Modelling Scenarios

Refinement

- Concept: A (Transition-)System:

- States were encoded by a schema, where the predicative part contains the system invariant

σ
$x_1:T_1; \dots ; x_n:T_n$
Inv

- Operations were encoded by a schema importing the state Δ via and Ξ operator.

op
$\Delta\sigma;$
$x? : S$
$y! : T$
...

op
$\Xi\sigma;$
$x? : S$
$y! : T$
...

Advanced Modelling Scenarios

Analysis (consistency,...)

- A Transitionsystem is consistent if:

- the set of initial states is non-empty:

$$\exists \sigma \in \text{Init}$$

- a state invariant is satisfiable:

$$\exists \sigma \in \text{Inv}$$

- all operations op are implementable:

$$\forall \sigma, i?. \text{PRE}(\sigma, i?) \rightarrow \exists \sigma'. (o!. \sigma, i?, \sigma', o!) \in op$$

where $\text{PRE}(\sigma, i?)$ is the **syntactic** precondition of op .

Advanced Modelling Scenarios

Analysis (consistency,...)

- Transitionsystem consistency is checks in HOL-Z by a number of “analytical statements”, top-level commands that generate proof-obligations wrt. the system.
 - Init-state non-emptiness: `gen_state_cc “ σ ”`
 - Invariant non-emptiness: `gen_state_cc “ σ ”`
 - Operation implementable: `gen_op_cc “op”`
- Proof-obligations can be referenced by the `po` command and discharged by conventional Isabelle proofs.

Advanced Modelling Scenarios

Refinement

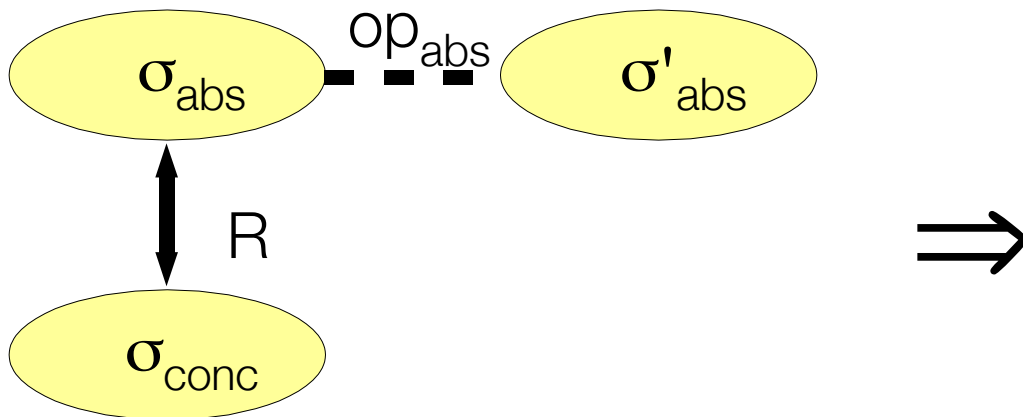
- Concept (Refinement Conditions):
 - Give abstraction relation $R : \mathbb{P}(\sigma_{\text{abs}} \times \sigma_{\text{conc}})$ relating concrete and abstract states
 - Init - Condition:

$$(\sigma_{\text{abs}}, \sigma_{\text{conc}}) \in R \rightarrow \sigma_{\text{abs}} \in \text{Init}_{\text{abs}} \rightarrow \sigma_{\text{conc}} \in \text{Init}_{\text{con}}$$

Advanced Modelling Scenarios

Refinement

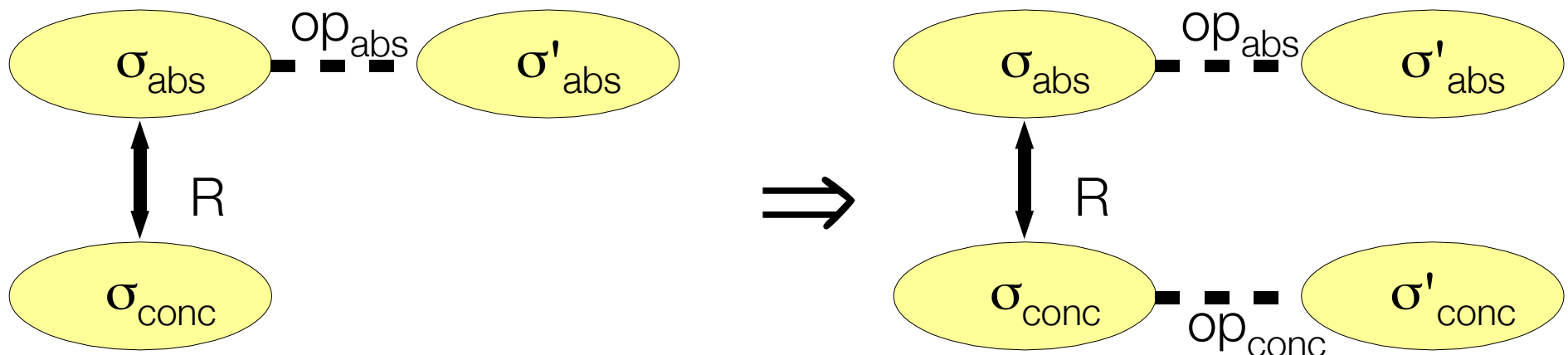
- Concept (Refinement Conditions):
 - Give abstraction relation $R : \mathbb{P}(\sigma_{\text{abs}} \times \sigma_{\text{conc}})$ relating concrete and abstract states
 - Init - Condition
 - Preserve Enabledness:



Advanced Modelling Scenarios

Refinement

- Concept (Refinement Conditions):
 - Give abstraction relation $R : \mathbb{P}(\sigma_{\text{abs}} \times \sigma_{\text{conc}})$ relating concrete and abstract states
 - Init - Condition
 - Preserve Enabledness:



Advanced Modelling Scenarios

Refinement

- Concept (Refinement Conditions):
 - Give abstraction relation $R : \mathbb{P}(\sigma_{\text{abs}} \times \sigma_{\text{conc}})$ relating concrete and abstract states
 - Init - Condition
 - Preserve Enabledness:

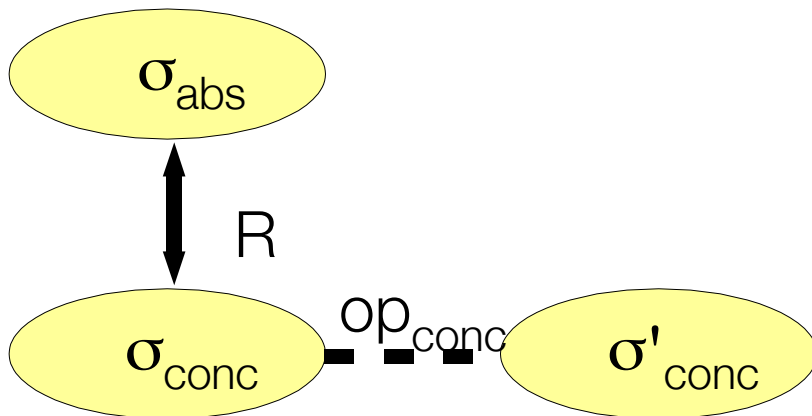
$$\forall \sigma_{\text{abs}} \in \text{dom}(\text{op}_{\text{abs}}), \sigma_{\text{conc}} \in \text{Inv}. (\sigma_{\text{abs}}, \sigma_{\text{conc}}) \in R \rightarrow \sigma_{\text{conc}} \in \text{dom}(\text{op}_{\text{conc}})$$

where $\text{dom}(S) \subseteq \text{Inv}$

Advanced Modelling Scenarios

Refinement

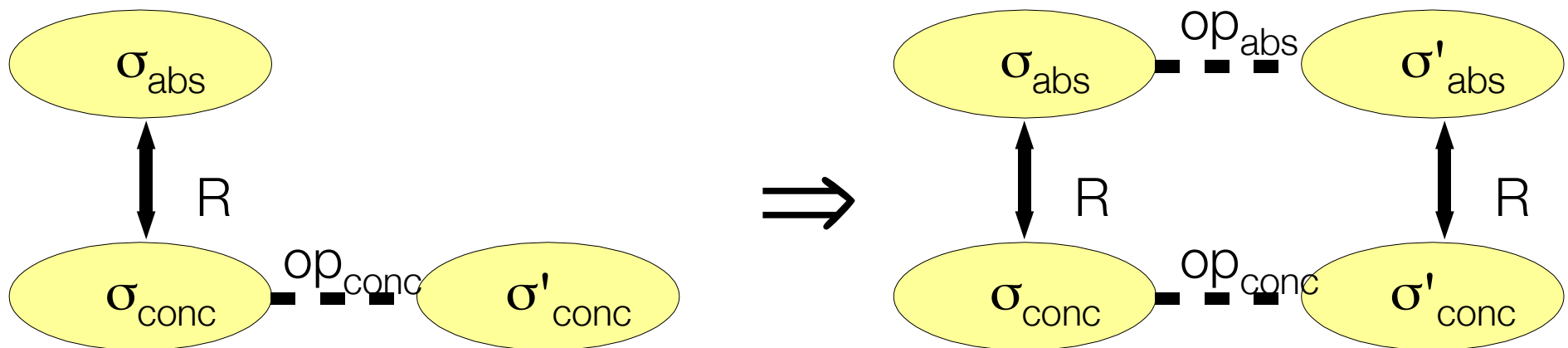
- Concept (Refinement Conditions):
 - Give abstraction relation $R : \mathbb{P}(\sigma_{\text{abs}} \times \sigma_{\text{conc}})$ relating concrete and abstract states
 - Init - Condition; Preserve Enabledness:
 - Refine



Advanced Modelling Scenarios

Refinement

- Concept (Refinement Conditions):
 - Give abstraction relation $R : \mathbb{P}(\sigma_{\text{abs}} \times \sigma_{\text{conc}})$ relating concrete and abstract states
 - Init - Condition; Preserve Enabledness:
 - Refine



Advanced Modelling Scenarios

Refinement

- Concept (Refinement Conditions):
 - Give abstraction relation $R : \mathbb{P}(\sigma_{\text{abs}} \times \sigma_{\text{conc}})$ relating concrete and abstract states
 - Init - Condition; Preserve Enabledness:
 - Refine

$$\begin{aligned} &\forall \sigma_{\text{abs}} \in \text{dom}(\text{op}_{\text{abs}}); \sigma_{\text{conc}}, \sigma'_{\text{conc}} \in \text{Inv}. \\ &\quad (\sigma_{\text{abs}}, \sigma_{\text{conc}}) \in R \wedge (\sigma_{\text{conc}}, \sigma'_{\text{conc}}) \in \text{op}_{\text{conc}} \\ &\quad \rightarrow \exists \sigma'_{\text{abs}} \in \text{Inv}. (\sigma_{\text{abs}}, \sigma'_{\text{abs}}) \in \text{op}_{\text{abs}} \wedge (\sigma'_{\text{abs}}, \sigma'_{\text{conc}}) \in R \end{aligned}$$

where $\text{dom}(S) \subseteq \text{Inv}$

Refinement Support in HOL-Z

- The HOL/ISAR command language is extended by HOL-Z specific top-level commands.

- declaring the refinement relation

```
set_abs "R"
```

- or

```
set_abs "R" [functional]
```

for a functional refinement setting (simpler !)

Refinement Support in HOL-Z

- The HOL/ISAR command language is extended by HOL-Z specific top-level commands.
 - analytical command for generating the init-condition (treated as HOL-Z proof-obligation)
`refine_init` “init_{abs}” “init_{conc}”
 - analytical command for generating the op-condition (treated as HOL-Z proof-obligation)
`refine_op` op_{abs} op_{conc}

Refinement Support in HOL-Z

- The HOL/ISAR command language is extended by HOL-Z specific commands:
 - bookkeeping commands:

show_po

list_po

check_po

check_po [except “po-class”]

Refinement Support in HOL-Z

- The HOL/ISAR command language is extended by HOL-Z specific commands:
 - discharge for proof obligations:

```
po <po-name>:  
  apply(method )  
  ...  
  apply(method )  
  discharged
```

DEMO

- *Advanced Modelling Scenarios*
Modeling Temporal Properties

-

More to come

Case Studies

- HOL-Z has been applied to several case studies:

- BirthdayBook

academic example stemming from Spivey's ZRM for a small data base system where a relations is refined by two arrays and a high-watermark

Contains sample proofs for analysis and refinement.

Case Studies

- HOL-Z has been applied to several case studies:

- DARMA

Client-Server Architecture for a Digital Signature Specification. Clients may login, logout, or authenticate a document in various sessions.

Design Document provided by industrial partner (Hitachi).

Contains advanced proofs.

Case Studies

- HOL-Z has been applied to several case studies:
 - CVS-Server

An abstract versioning system (with role-based access control security model inside) is refined towards a concrete configuration over a CVS-managed repository in a POSIX – UNIX filesystem.

Conclusion

- Z is very similar to HOL.
- This leads to a cleaner understanding
- and useable, taylorred proof-tools like
HOL-Z !!!

Bibliography

- David Basin and Hironobu Kuruma and Kunihiro Miyazaki and Kazuo Takaragi and Burkhart Wolff. **Verifying a signature architecture: a comparative case study**. In *Formal Aspects of Computing*, 19 (1), pages 63-91, 2007.
- David Basin and Hironobu Kuruma and Shin Nakajima and Burkhart Wolff. **The Z Specification Language and the Proof Environment Isabelle/HOL-Z**. In *Computer Software - Journal of the Japanese Society for Software Science and Technology*, 24 (2), pages 21-26, 2007. In Japanese.
- Makarius Wenzel and Burkhart Wolff. Building Formal Method Tools in the Isabelle/Isar Framework. In *TPHOLs 2007*. LNCS 4732 Springer-Verlag, 2007.
- David Basin and Hironobu Kuruma and Kazuo Takaragi and Burkhart Wolff. **Verification of a Signature Architecture with HOL-Z**. In *Formal Methods 2005*. LNCS 3582 Springer Verlag, 2005.
- Achim D. Brucker and Burkhart Wolff. A Verification Approach for Applied System Security. In *International Journal on Software Tools for Technology Transfer (STTT)*, 7 (3), pages 233-247, 2005.
- Achim D. Brucker and Frank Rittinger and Burkhart Wolff. **HOL-Z 2.0: A Proof Environment for Z-Specifications**. In *Journal of Universal Computer Science*, 9 (2), pages 152-172, 2003.
- Christoph Lüth and Einar W. Karlsen and Kolyang and Stefan Westmeier and Burkhart Wolff. HOL-Z in the UniForM-Workbench - a Case Study in Tool Integration for Z. In *11. International Conference of Z Users ZUM'98*. LNCS 1493. Springer Verlag, 1998.
- Kolyang and T. Santen and B. Wolff. A Structure Preserving Encoding of Z in Isabelle/HOL. In *Theorem Proving in Higher Order Logics - 9th International Conference*. LNCS 1125 Springer Verlag, 1996.
- Tobias Nipkow, Lawrence C. Paulson, Markus Wenzel: **Isabelle/HOL A Proof Assistant for Higher-Order Logic**. LNCS 2283. 2005.
- Jim Woodcock, Jim Davies: **UDavid Basin SING Z: SPECIFICATION, REFINEMENT AND PROOF**. Prentice Hall ISBN: 0-13-948472-8, May, 1996.

Sources

- available under:

<http://www.brucker.ch/projects/hol-z/>

(based on Isabelle 2005)

