Analysing Z Specifications with HOL-Z

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

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

Abstract

The increasing complexity of todays 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 auto–mation 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    (CadiZ (experimental), Z/EVES (outdated),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?

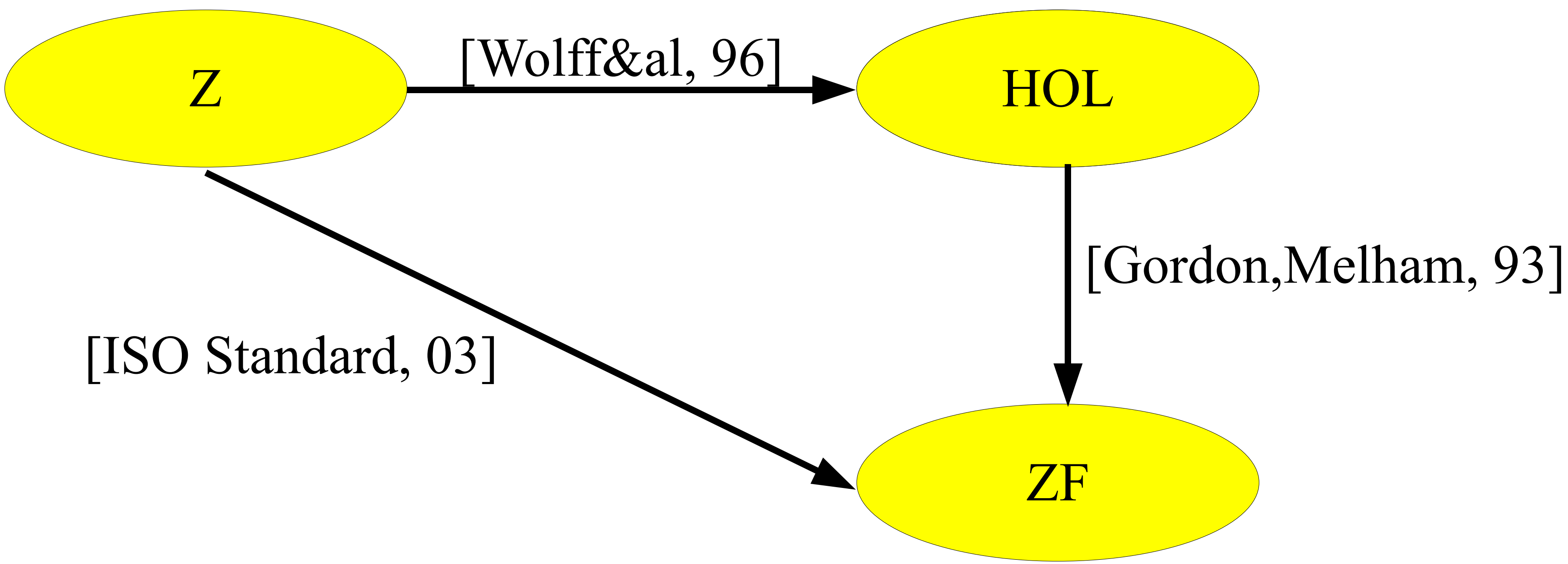
- Z Semantics via Embedding in Higher-Order Logic (HOL)

  - 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:

$$\begin{aligned}\tau := {} & \mathbb{Z} \\ & \mid \tau \times \cdots \times \tau \\ & \mid \langle\!|\ tag_1 \leadsto \tau, \ldots, tag_n \leadsto \tau |\!\rangle \\ & \mid \mathbb{P}(\tau)\end{aligned}$$

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

Foundations: Z, HOL and Z in HOL Semantics

- HOL **has**:
  - declaration principles:

arities χ :: “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 : "$s = t \Longrightarrow t = s$"
mp : "$P \Longrightarrow P \rightarrow Q \Longrightarrow Q$"
subst: "$[\![s=t;\ P\ s]\!] \Longrightarrow P\ t$"
...

where $\Longrightarrow$ is the meta-implication. We write

$[\![A_1;\ldots;A_n]\!] \Longrightarrow B$ **for** $A_1 \Longrightarrow \ldots \Longrightarrow A_n \Longrightarrow B$.

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 : "$\forall \equiv \lambda P. (P = \lambda x.True)$"

(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 α χ "..."
    consts Abs_χ :: τ set $\longrightarrow$ α χ
    Rep_χ :: α χ $\longrightarrow$ τ set
    axioms Abs_χ_inverse : "Abs_χ (Rep_χ x) = x"
    Rep_χ_inverse : "E x $\Rightarrow$
    Rep_χ (Abs_χ x) = x"

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 priciples), 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, ...
  - pairs: $(a_1, \ldots, a_n)$, pattern-abstractions: $\lambda(a_1, \ldots, a_n).E$
  - records: $\langle\!|tag_1 \rightsquigarrow E, \ldots, tag_n \rightsquigarrow E|\!\rangle$

Foundations: Z, HOL and Z in HOL Semantics

- (typed) Expressions E in Z are:
  - arithmetic: 1,2,3, a + b, a / b, ...

    semantics: Library Theory Arith with type type int = $\mathbb{Z}$
  - pairs:$(a_1,\ldots,a_n)$, pattern-abstractions: $\lambda(a_1, \ldots, a_n).E$

    semantics: Library Theory Pair with type $\_\times\_$
  - records: $\langle\!|tag_1 \rightsquigarrow \tau , . . . , tag_n \rightsquigarrow \tau|\!\rangle$

    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 : $\mathbb{N}, \mathbb{Z}$ ($:: \mathbb{P}(\mathbb{Z})$!!!)
  - 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** : $\mathbb{N}, \mathbb{Z}$ (:: $\mathbb{P}(\mathbb{Z})$!!!)

    constdefs $\mathbb{N} \equiv \{a :: int \mid 0 \leq a\}, \ldots$
    $\mathbb{Z} \equiv \{a :: int \mid True\}, \ldots$
    (HOL comprehension!)

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

    constdefs "$f\,´\,S \equiv \{a :: int \mid \exists y.\ a = f\ y\}$"
    "$\{a \in B \mid P(a) \bullet f(a)\} \equiv$
    $f\,´\,\{x \mid a \in B \wedge P(a)\}$

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: $\mathbb{F}(A)$, $\mathbb{P}(A)$ ($\mathbb{P}$ :: $\mathbb{P}(\mathbb{P}(A) \times \mathbb{P}\mathbb{P}(A))$!!!)

Foundations: Z, HOL and Z in HOL Syntax

- (typed) Expressions E in Z are:
  - **set predicates:** $A \subseteq B$

    constdefs $A \subseteq B \equiv \forall a.\ a \in A \longrightarrow a \in B$
  - **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:** $\mathbb{P}(A)$ $(\mathbb{P} :: \mathbb{P}(\mathbb{P}(A) \times \mathbb{P}\mathbb{P}(A))$!!! $)$, $\mathbb{F}(A)$

    constdefs $\mathbb{P}(A) \equiv \{B \mid B \subseteq A\}$

    $\mathbb{F}(A) \equiv \{B \mid B \subseteq A \wedge \text{finite } B\}$

Foundations: Z, HOL and Z in HOL Semantics

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

$A \leftrightarrow B$	: relation from A to B
$A \pfun B$	: partial function from A to B
$A \longrightarrow B$	: total function from A to B
$A \pinj B$	: bijective partial function from A to B
$A \rightarrowtail\!\!\!\!\twoheadrightarrow B$	: total finite bijection from A to B
$A \psurj B$	: partial finite surjection from A to B
...	

Foundations: Z, HOL and Z in HOL Syntax

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

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

$$A ⇸ B \equiv \{r \in A \longleftrightarrow B \mid \forall a \in \mathrm{dom}\, r. x,y \in B. (a,x) \in r \wedge (a,y) \in r \longrightarrow x = y\}$$

$$A \longrightarrow B \equiv \ldots$$

$$A ⤔ B \equiv \ldots$$

$$A \rightarrowtail B \equiv \ldots$$

$$A ⤀ B \equiv \ldots$$

...

Foundations: Z, HOL and Z in HOL Syntax

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

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)

    $$\begin{array}{|l} \_ \geq \_ : \mathbb{P}\,(date \times date) \\ \hline total\_ordering\,(\_ \geq \_) \end{array}$$

  - axiomatic definitions (2)

    $$x \equiv E$$

Foundations: Z, HOL and Z in HOL Semantics

- Toplevel-Constructs
  - abstract types

    arities book :: "..." ...

  - axiomatic definitions (1)

    consts $\_ \geq \_ : (\mathrm{date} \times \mathrm{date})$ set
    axiom order_axdef :
    "$\_ \geq \_ \in \mathbb{P}(\mathrm{date} \times \mathrm{date}) \wedge$
    total_ordering $(\_ \geq \_)$"

  - axiomatic definitions (2)

    $x \equiv E$

Foundations: Z, HOL and Z in HOL Syntax

- Toplevel-Constructs
  - schema declarations:

```
    A
 ________________________________________
| d1 : T1
|    ...                  --  declaration part
| dn : Tn
|_______________________
|
| P(d1,..,dn)             --  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:

$$
A \quad \equiv \quad
\begin{array}{|l}
\hline
d_1 : T_1 \\
\quad \ldots \qquad\qquad\qquad \text{-- declaration part} \\
d_n : T_n \\
\hline
P(d_1,..,d_n) \qquad\qquad \text{-- predicative part} \\
\\
\hline
\end{array}
$$

Foundations: Z, HOL and Z in HOL Semantics

- Toplevel-Constructs
  - schema declarations:

    ISO-STANDARD:

$$A \quad \equiv \quad \{ d_1 : T_1;\ldots; d_n : T_n \mid P(d_1,..,d_n) \bullet \langle\!| d_1 \leadsto d_1,.., d_n \leadsto d_1 |\!\rangle \}$$

From Foundations to Pragmatics Semantics

- Toplevel-Constructs
  - schema declarations:

    EQUIVALENTLY :
    SCHEMAS AS FUNCTIONS:

$$A \equiv \lambda(a_1,\ldots,a_n) \bullet a_1 \in T_1 \wedge \ldots \wedge a_n \in T_n$$
$$\wedge\ P(a_1,\ldots,a_n)$$

From Foundations to Pragmatics Semantics

- Toplevel-Constructs
  - schema declarations:

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

$$A \equiv \mathrm{SB}\ \text{“}a_1\text{“} \rightsquigarrow a_1, \ldots, \text{“}a_n\text{“} \rightsquigarrow a_n \bullet \quad a_1 \in T_1 \wedge \ldots \wedge a_n \in T_n \wedge P(a_1, \ldots, a_n)$$

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

$B;$

$z \;: seq(A)$

$R(x_1, x_2, x'_1, x', z)$

Foundations: Z, HOL and Z in HOL
Syntax

- Toplevel-Constructs: schemas with imports:

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

B

$A;\ A';$
$x_2\ :\ T$

$Q(x_1,x_2,x'_1,x'_2)$

C

$B;$
$z\ :\ seq(A)$

$R(x_1,x_2,x'_1,x',\ z)$

Foundations: Z, HOL and Z in HOL Syntax

- Toplevel-Constructs: schemas with imports:

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

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

C

$$\begin{array}{|l}
B; \\
z\ :\ seq(A) \\
\hline
R(x_1,x_2,x'_1,x', z)
\end{array}$$

Foundations: Z, HOL and Z in HOL Syntax

- Toplevel-Constructs: schemas with imports:

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

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

$$C \equiv \lambda(x_1,x_2,x'_1 x'_2 z) \bullet B(x_1,x_2,x'_1,x'_2) \wedge x_2 \in seq(asSet\ A) \wedge R(x_1,x_2,x'_1,x'_2, z)$$

Foundations: Z, HOL and Z in HOL Syntax

- Toplevel-Constructs: schemas with imports:

$$A \equiv \; SB\ \text{“}x_1\text{”} \rightsquigarrow x_1, \text{”}x_2\text{”} \rightsquigarrow x_2 \bullet$$
$$x_1 \in S_1 \wedge x_2 \in S_2$$
$$\wedge\ P(x_1, x_2)$$

$$B \equiv SB\ \text{“}x_1\text{”} \rightsquigarrow x_1, \text{”}x_2\text{”} \rightsquigarrow x_2$$
$$\text{“}x'_1\text{”} \rightsquigarrow x'_1, \text{”}x'_2\text{”} \rightsquigarrow x'_2 \bullet$$
$$A(x_1, x_2) \wedge A(x'_1, x'_2) \wedge$$
$$x_2 \in T \wedge$$
$$Q(x_1, x_2, x'_1, x'_2)$$

$$C \equiv SB\ \text{“}x_1\text{”} \rightsquigarrow x_1, \text{”}x_2\text{”} \rightsquigarrow x_2\ \text{“}x'_1\text{”} \rightsquigarrow x'_1, \text{”}x'_2\text{”} \rightsquigarrow x'_2\ \text{”}z\text{”} \rightsquigarrow z \bullet$$
$$B(x_1, x_2, x'_1, x'_2) \wedge$$
$$x_2 \in seq(asSet\ A) \wedge$$
$$R(x_1, x_2, x'_1, x'_2,\ z)$$

Foundations: Z, HOL and Z in HOL Syntax

- Toplevel-Constructs: Schema Calculus
  - schema decoration: S'

    ΔS

    ΞS
  - schema expressions: $A \wedge B$

    $S \Leftrightarrow S$, $S \vee S$,

    $S \rightarrow S$, $\neg S$,...
  - schema quantification $\forall S \bullet S$, $\exists S \bullet S$, ...

Foundations: Z, HOL and Z in HOL Semantics

- Toplevel-Constructs: Schema Calculus
  - schema decoration: S' renaming signature (as bef.)

    $\Delta S \equiv S \wedge S'$

    $\Xi S \equiv \Delta S \wedge (x_1 = x'_1 \wedge \ldots \wedge x_n = x'_n)$
  - schema expressions: $A \wedge B \equiv \lambda(x_1 \ldots x_n).A(x_{i1},\ldots,x_{im}) \wedge B(x_{j1},\ldots,x_{jm})$

    $S \Leftrightarrow S, S \vee S, \quad \ldots$

    $S \rightarrow S, \neg S, \ldots \quad \ldots$
  - schema quantification $\forall S \bullet S, \exists S \bullet S, \ldots$

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 prediates
  - HOL-Z: schemas were represented as characteristic functions – the structure is preserved (no unfolding, flattening, etc, ...)

The HOL-Z System

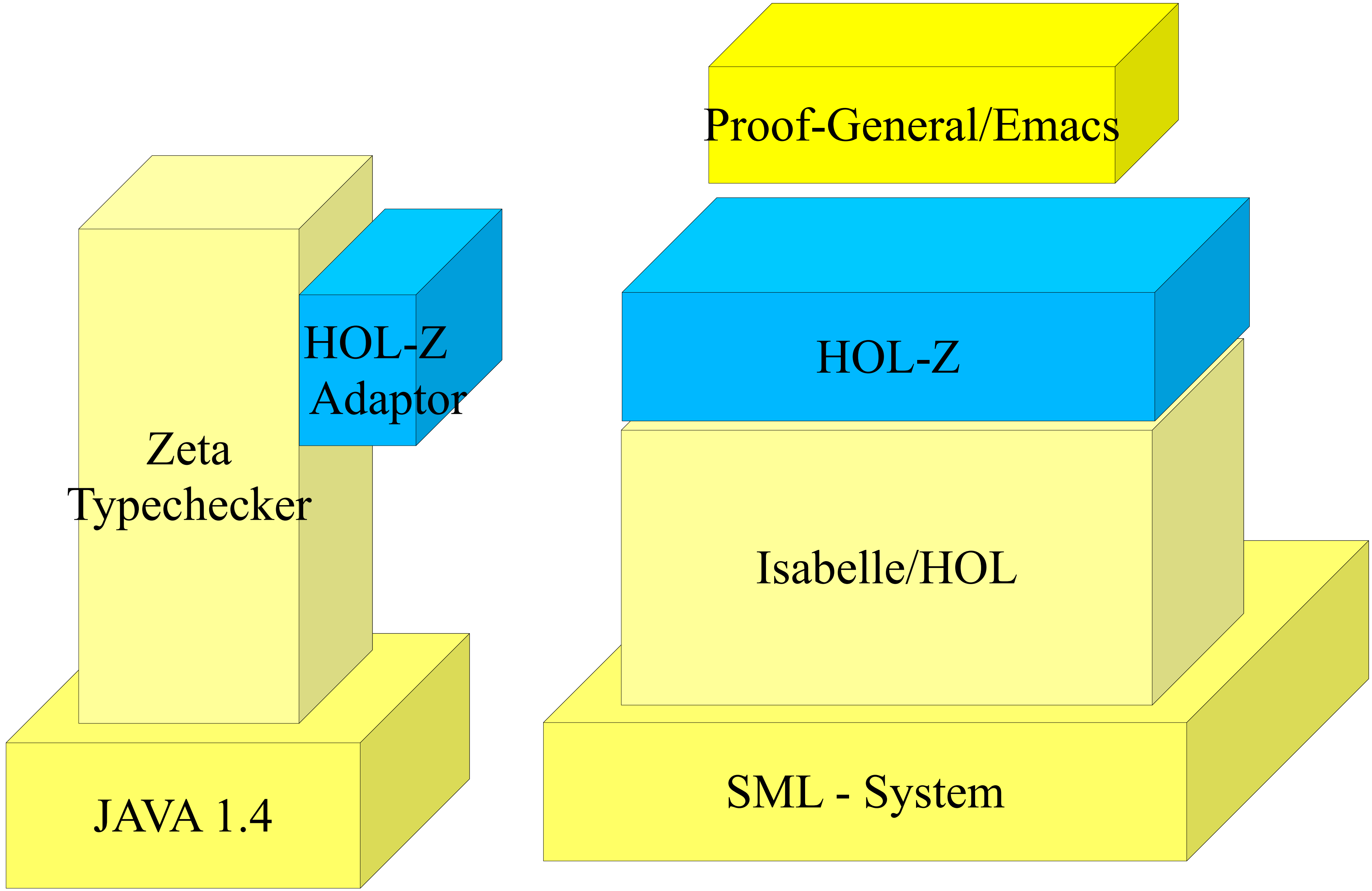
- Based on an advanced, interactive proof-environment: Isabelle/HOL
- Conceived as Plugin into the Isabelle/ISAR architecture
- Provides Parser, conservative semantic theories, own proof procedures
- A refinement package to support top-down development

The HOL-Z System

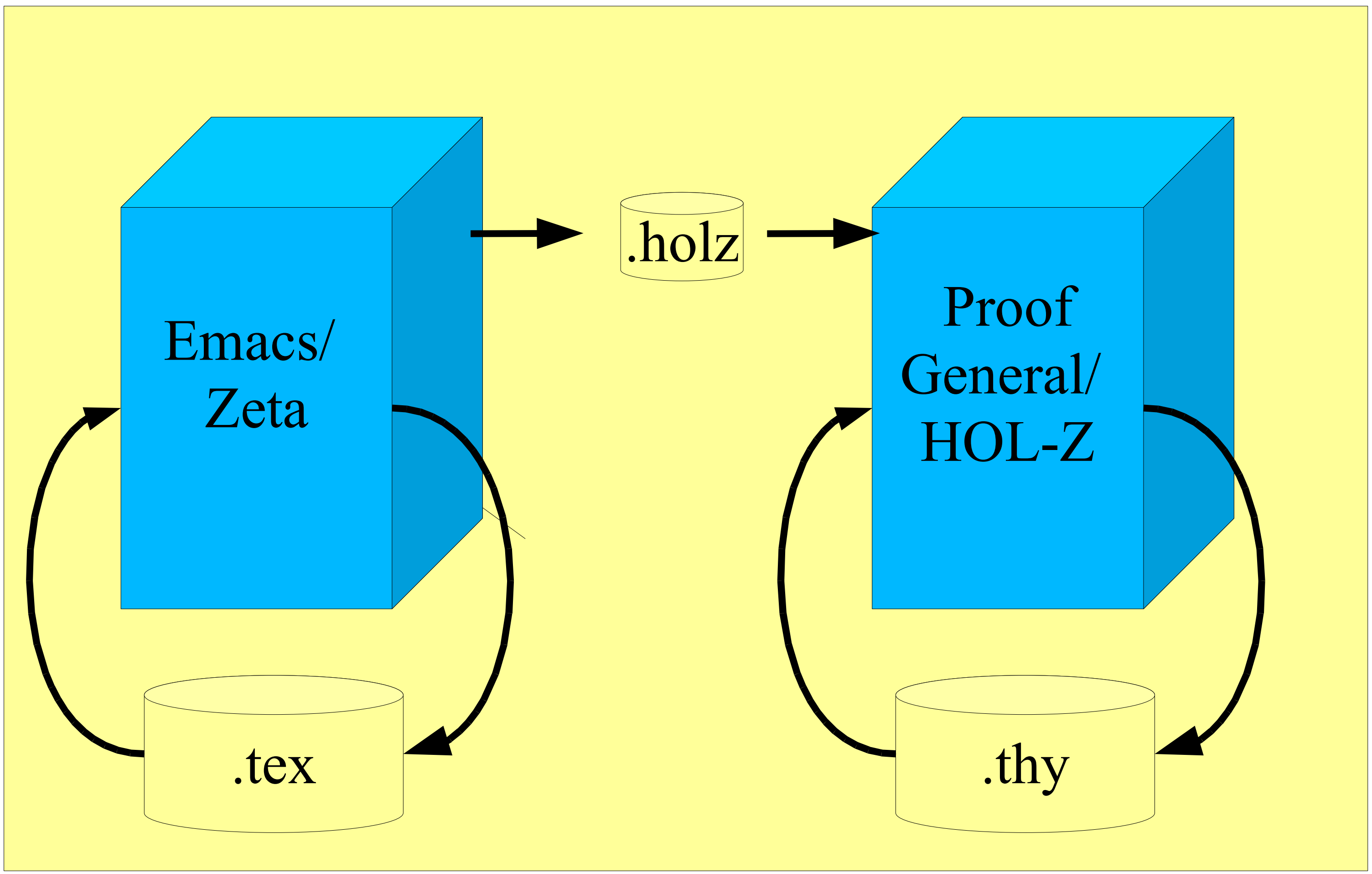
- The HOL-Z System Architecture
- The Workflow

Underlying Assumption:
Designer and Proof-Engineer not necessarily the same person !

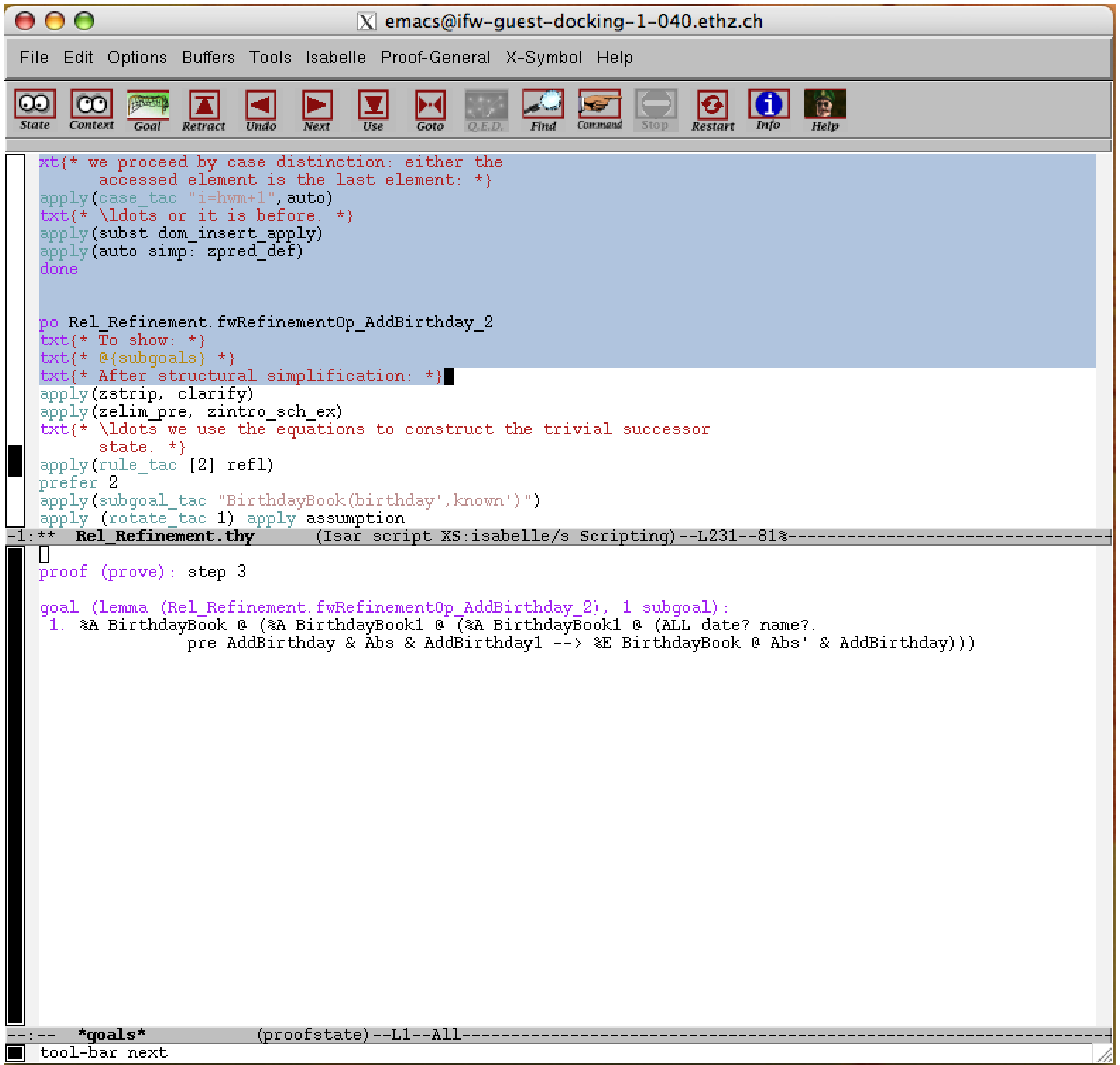
The HOL-Z System Component View



The HOL-Z System
Workflow View



The HOL-Z System
ProofGeneral



checked area
(non-editible)

unchecked area
(editible)

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	
$\neg$	~, not, \<lnot>	\lnot	unary
pre	PRE	\pre	unary
$\wedge$	&,\<land>, /\	\land	left
$\vee$	|, \<lor>, \/	\lor	left
$\longrightarrow$	-->, \<implies>	\implies	right
$\Leftrightarrow$	=	\iff	left
$\forall$	!, \<forall>	\forall	
$\exists$	?,\<exists>	\exists	
$\upharpoonright$	project	\project	left
\	hide	\hide	left
;	not supported	\semi	left
>>	not supported	\pipe	left
if	if	\IF	
then	then	\THEN	
else	else	\ELSE	
let	let	\LET	

The HOL-Z System
Z LaTeX : Sets

• Math	HOL-Z	ZETA	
ℙ	Pow, \<power>	\power	pregen
𝔽	Fin, \<finset>	\finset	pregen
∅	{}	\emptyset	word
#	card	\#	word
∩	Int, \<cap>	\cap	inop 4
∪	Un, \<cup>	\cup	inop 4
-	-	\setminus	inop 3
⊆	<=, \<subseteq>	\subseteq	inrel
⊂	<	\subset	inrel
=	=	= =	inrel
≠	~=	\neq	inrel
∈	:, \<in>	\in	inrel
∉	~: \<notin>	\notin	inrel

The HOL-Z System
Z LaTeX : Relations and Functions

<table>
<tr><th>• Math</th><th>HOL-Z</th><th>ZETA</th><th></th></tr>
<tr><td>$\mapsto$</td><td>|->,\<mapsto></td><td>\mapsto</td><td>inop 1</td></tr>
<tr><td>$\times$</td><td>*</td><td>\cross</td><td>inop 1</td></tr>
<tr><td>dom</td><td>dom</td><td>\dom</td><td>word</td></tr>
<tr><td>ran</td><td>ran</td><td>\ran</td><td>word</td></tr>
<tr><td>$\circ$</td><td>o</td><td>\circ</td><td>inop 4</td></tr>
<tr><td>;</td><td>\<semi></td><td>\comp</td><td>inop 5</td></tr>
<tr><td>$\oplus$</td><td>(+),\<oplus></td><td>\oplus</td><td>inop 5</td></tr>
<tr><td>$\sim$</td><td>not supported</td><td>\inv</td><td>postop</td></tr>
<tr><td>$\lhd$</td><td><|, \<dres></td><td>\dres</td><td>inop 6</td></tr>
<tr><td>$\rhd$</td><td>|>, \<rres></td><td>\rres</td><td>inop 6</td></tr>
<tr><td>⩤</td><td><-|, \<ndres></td><td>\ndres</td><td>inop 6</td></tr>
<tr><td>⩥</td><td>|->, \<nrres></td><td>\nrres</td><td>inop 6</td></tr>
<tr><td>+</td><td>not supported</td><td>\plus</td><td>postop</td></tr>
<tr><td>*</td><td>not supported</td><td>\star</td><td>postop</td></tr>
</table>

The HOL-Z System
Z LaTeX : Relations and Functions

Math	HOL-Z	ZETA	
f(.x.)	f%^x,f\<rappll>x\<rapplr>	f(x)	(* relational application *)
*	\<star>	\star	postop
↔	<->,\<rel>	\rel	ingen
⇸	-|->, \<pfun>	\pfun	ingen
→	--->, \<fun>	\fun	ingen
⤔	>-|->,\<pinj>	\pinj	ingen
↣	>-->, \<inj>	\inj	ingen
⤀	-|->>, \<psurf>	\psurj	ingen
↠	-->>,\<surj>	\surj	ingen
⤖	>-->,\<bij>	\bij	ingen
⇻	-||->\<ffun>	\ffun	ingen
⤕	>-||-> ,\<finj>	\finj	ingen
id	id	\id	word
⦇	\<limg>	\limg	-
⦈	\<rimg>	\rimg	-

The HOL-Z System

Z LaTeX : Integers

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

The HOL-Z System
Z LaTeX : Integers

Math	HOL-Z	ZETA	
$\langle$		\langle	-
$\rangle$		\rangle	-
seq	seq	\seq	pregen
iseq	iseq	\iseq	pregen
in	seqin	\inseq	inrel
$\frown$	seqconcat	\cat	inop 3
prefix	prefix	\prefix	inrel
suffix	suffix	\suffix	inrel
$\upharpoonright$	|\`	\filter	word
??	´|	\extract	word

The HOL-Z System
Z LaTeX : Toplevel

- Math HOL-Z ZETA
  - Modules

    .thy - structure \zsection[name ... name]{name}

  - Type Definitions

    [type] arities type ... \begin{zed}
    [type]
    \end{zed}

  - Axiomatic Definitions

Math	HOL-Z	ZETA
$\dfrac{f : E}{P(f)}$	axiom <name> : ...	\begin{axdef} f : E \where pred(f) \end{axdef}

The HOL-Z System
Z LaTeX : Toplevel

- Math HOL-Z ZETA
  - Schema Definitions

A

$x_1 : S_1$

$x_2 : S_2$

$P(x_1,x_2)$

constdef <name>: ...

```
\begin[A]{schema}
    x_1 : S1;
    x_2 : S2
\where
    P(x_1,x_2)
\end{axdef}
```

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

- Hierachical 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

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

zturnstyleE

Theorem Proving in HOL-Z

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

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

zschallI

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

zschallE

Theorem Proving in HOL-Z

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

$$\frac{S(\langle y\rangle, \langle ?x'\rangle, \langle ?x!\rangle)}{\mathbf{pre}\ S}\ (*)$$

zpreI

$$\frac{\mathbf{pre}\ S \qquad \begin{array}{c}[S(\langle y\rangle,\langle x'\rangle,\langle x!\rangle)]\\ \bigwedge \langle x'\rangle \langle x!\rangle.\ \ R\end{array}}{R}\ (*)$$

zpreE

Theorem Proving in HOL-Z

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

$$\frac{S(\langle ?x \rangle) \qquad T(\langle ?x \rangle \langle y \rangle)}{\exists\ S \bullet T}$$

zschexI

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

zschexE

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) $x_1,\ldots,x_n$
  - $\langle x \rangle \langle y \rangle$ denotes the concatenation of these vectors
  - $\langle \underset{\sim}{x} \rangle$ denotes a permutation of a vector
  - (*) stands for the proviso: $\langle y \rangle$ is a vector of free variables (not occuring 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:

$$sys_1 \rightsquigarrow sys_2 \rightsquigarrow \ldots \rightsquigarrow sys_n$$

  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

$$
\begin{array}{|l}
\multicolumn{1}{l}{\quad \sigma} \\
\hline
x_1 : T_1 ; \ldots ; x_n : T_n \\
\hline
Inv \\
\hline
\end{array}
$$

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

$$
\begin{array}{|l}
\multicolumn{1}{l}{\quad op} \\
\hline
\Delta\sigma; \\
x? \; : S \\
y! \; : T \\
\hline
\; \ldots \\
\hline
\end{array}
\qquad\qquad
\begin{array}{|l}
\multicolumn{1}{l}{\quad op} \\
\hline
\Xi\sigma; \\
x? \; : S \\
y! \; : T \\
\hline
\; \ldots \\
\hline
\end{array}
$$

Advanced Modelling Scenarios Analysis (consistency,...)

- A Transitionsystem is consistent if:
  - the set of initial states is non-empty:

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

  - a state invariant is satisfiable:

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

  - all operations op are implementable:

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

where $\mathrm{PRE}(\sigma, i?)$ 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-emptyness: gen_state_cc "σ"
  - Invariant non-emptyness: 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.

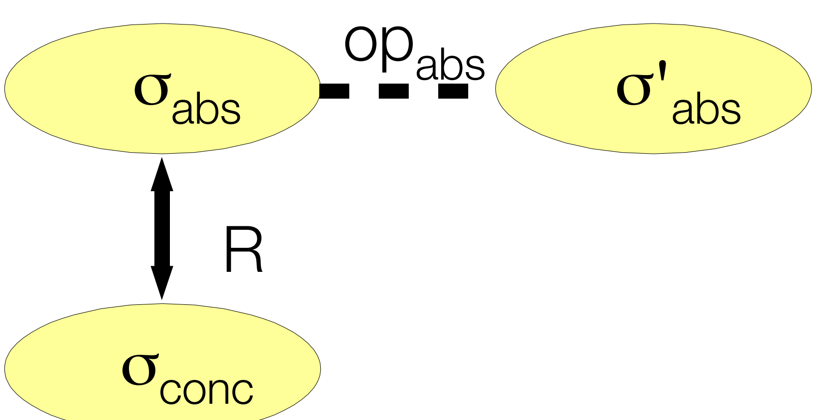
Advanced Modelling Scenarios
Refinement

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

$$(\sigma_{abs}, \sigma_{conc}) \in R \rightarrow \sigma_{abs} \in Init_{abs} \rightarrow \sigma_{conc} \in Init_{con}$$

Advanced Modelling Scenarios
Refinement

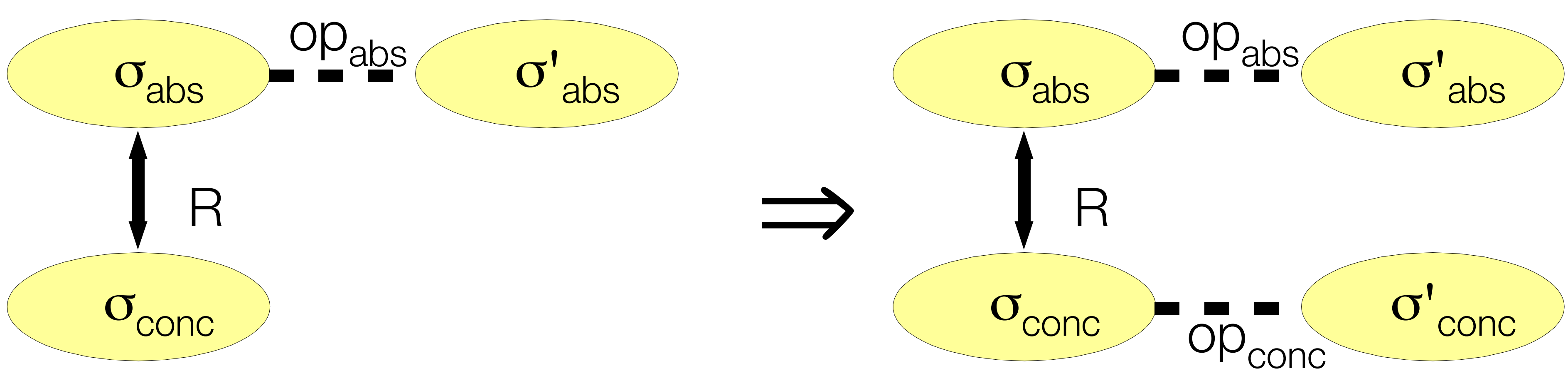
- Concept (Refinement Conditions):
  - Give abstraction relation $R : \mathbb{P}(\sigma_{abs} \times \sigma_{conc})$ relating concrete and abstract states
  - Init – Condition
  - Preserve Enabledness:



$\Rightarrow$

Advanced Modelling Scenarios
Refinement

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



Advanced Modelling Scenarios
Refinement

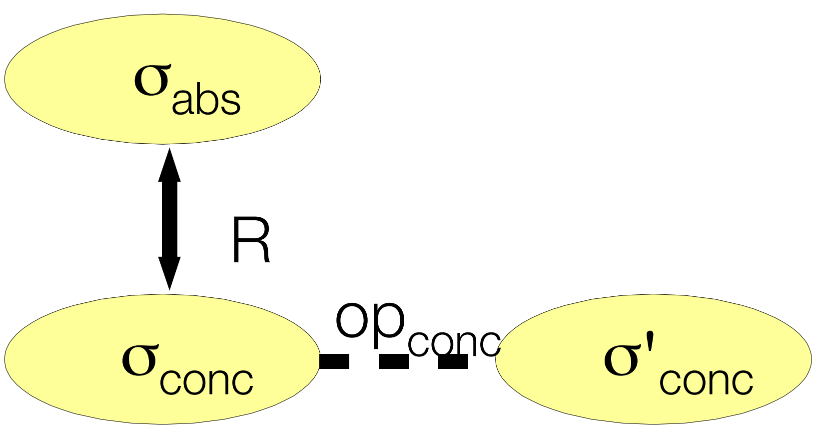
- Concept (Refinement Conditions):
  - Give abstraction relation $R : \mathbb{P}(\sigma_{abs} \times \sigma_{conc})$ relating concrete and abstract states
  - Init – Condition
  - Preserve Enabledness:

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

where $dom(S) \subseteq Inv$

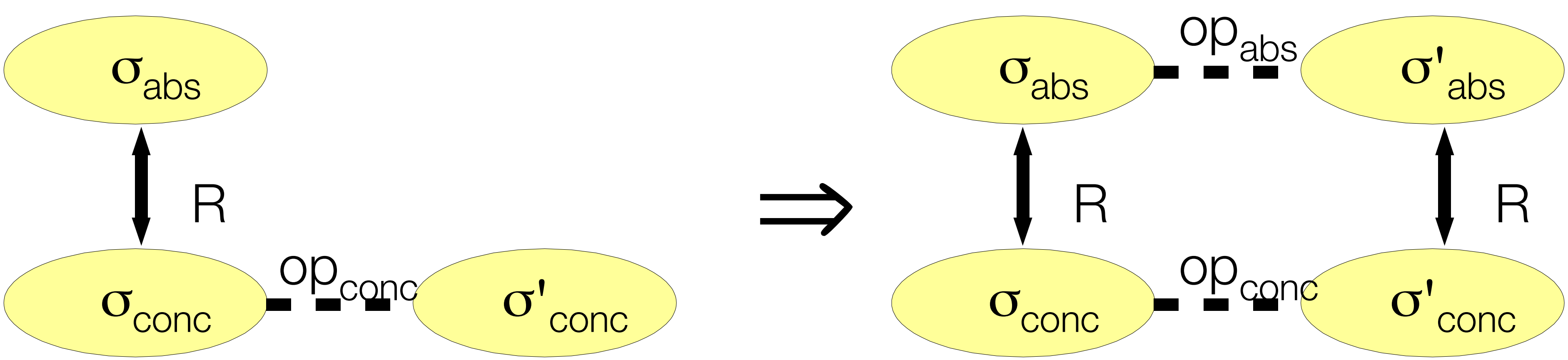
Advanced Modelling Scenarios
Refinement

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



Advanced Modelling Scenarios
Refinement

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



Advanced Modelling Scenarios
Refinement

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

$$\forall \sigma_{abs} \in dom(op_{abs});\ \sigma_{conc}, \sigma'_{conc} \in Inv.$$
$$(\sigma_{abs}, \sigma_{conc}) \in R \wedge (\sigma_{conc}, \sigma'_{conc}) \in op_{conc}$$
$$\rightarrow \exists \sigma'_{abs} \in Inv.\ (\sigma_{abs}, \sigma'_{abs}) \in op_{abs} \wedge (\sigma'_{abs}, \sigma'_{conc}) \in R$$

where $dom(S) \subseteq Inv$

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 init-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 Dignital 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 configutation 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, taylored proof-tools like HOL-Z !!!

Bibliography

- David Basin and Hironobu Kuruma and Kunihiko 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 1125Springer 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 BasinSING Z: SPECIFICATION, REFINEMENT AND PROOF. **Prentice Hall** ISBN: 0-13-948472-8, May, 1996.

Sources

- avaibable under:

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

  (based on Isabelle 2005)

Symbols

- ∅∈⊆∪∩ { | } × ∘ 𝔽ℙℕℤ
- ⦉⦊ ⟦ ⟧ ⦅⦆ ⦃ ⦄ ⟨⟩ 〈 〉 ⟪ ⟫ ⦃⦄⦅⦆⦇⦈⦉⦊⦑⦒⟮ ⟯⟦⟧⦃⦄
- ∀∃⊢⇔∨∧⟶ ¬≡
- ⟶⟷ ⟶⤔⇸⤖⇸⟼↣⤀⤀⤔⤕⇝⋮⟹⟼⟾≼≽⋮↪⇏⇹⇼
- λμτπαΘθΞΔ ≼ ⊗∧⊕ = •⊴ ⊵⊲ ▷ ⊕ × ∘
- ≤≥