Applying SMT Solvers to the Test Template Framework

Maximiliano Cristiá[1] and Claudia Frydman[2]

[1]CIFASIS and UNR, Rosario, Argentina
[2]CIFASIS-LSIS and AMU, Marseille, France

cristia@cifasis-conicet.gov.ar

MBT 2012 – Tallin, Estonia – March, 2012

The Test Template Framework and Fastest

The Test Template Framework (TTF)

- MBT method for Z specifications.
- Generates test specifications (test templates) and test cases.
- Unit testing.

P. Stocks and D. Carrington, "A Framework for Specification-Based Testing" in IEEE TOSE 1996.

Fastest

- It is the only existing implementation of the TTF.
- Tool support for all steps up to test case generation.
- Now it partially supports test case refinement.
- Available at `www.fceia.unr.edu.ar/~mcristia`.

Cristiá and Rodríguez Monetti, "Implementing and Applying the Stocks-Carrington Framework for MBT", in ICFEM 2009.

Difficulties with the TTF

> *The ideas of test templates and of automatic partition of Z schema have been proposed several years ago. Their actual application to systems of industrial size was problematic due to the lack of efficient and adequate constraint solvers and theorem provers. Given the progresses recently achieved for these kinds of tools, these techniques are worth revisiting . . .*

Ana Cavalcanti and Marie-Claude Gaudel (2011): *Testing for refinement in Circus*. Acta Informatica, 48(2).

The Z notation

- First-order logic.
- Zermelo-Fraenkel set theory.
- Rich mathematical toolkit.
  - Binary relations, partial functions, sequences, etc.
- Schema calculus (structures large specifications).
- Emphasis on partial functions.

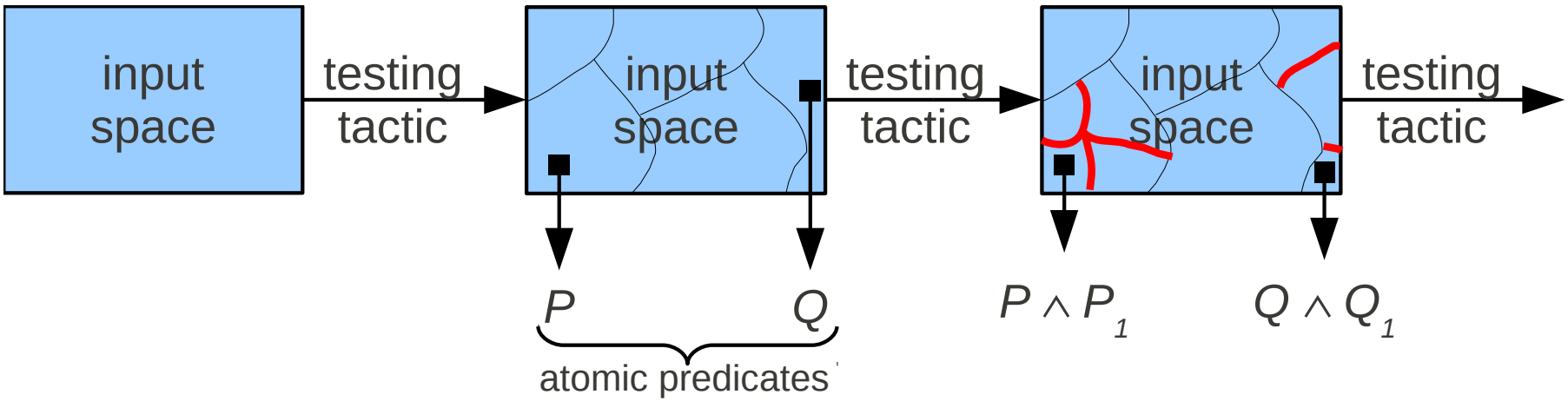
NewClientOk

$$\Delta SavingsAccounts$$
$$u? : UID;\ name? : NAME;\ n? : AN$$

$$u? \notin \operatorname{dom} clients$$
$$n? \notin \operatorname{dom} balances$$
$$clients' = clients \cup \{u? \mapsto name?\}$$
$$balances' = balances \cup \{n? \mapsto 0\}$$
$$owners' = owners \cup \{u? \mapsto n?\}$$

Test templates
(test specifications, test classes, test objectives, . . .)

Given a Z operation, take its input space and get a partition of it.



Each member of a partition is given by a Z predicate (i.e., first-order logic and Zermelo-Fraenkel set theory).

Testing tactics

DNF, standard partitions, sub-domain propagation, etc.

A typical test template

$$
\begin{array}{|l}
\hline
\mathit{RetrieveEData}^{SP}_{24} \\
\mathit{mem} : \operatorname{seq} \mathit{MDATA} \\
m : \mathbb{N} \\
d? : \operatorname{seq} \mathit{MDATA} \\
\hline
43 < m + \# d? \\
\mathit{mem} \neq \emptyset \\
\{ i : 1 \mathinner{..} \# d? \bullet m + i \mapsto d? \ i \} \neq \emptyset \\
\operatorname{dom} \mathit{mem} \cap \operatorname{dom} \{ i : 1 \mathinner{..} \# d? \bullet m + i \mapsto d? \ i \} \neq \emptyset \\
\neg \ \operatorname{dom} \{ i : 1 \mathinner{..} \# d? \bullet m + i \mapsto d? \ i \} \subseteq \operatorname{dom} \mathit{mem} \\
\neg \ \operatorname{dom} \mathit{mem} \subseteq \operatorname{dom} \{ i : 1 \mathinner{..} \# d? \bullet m + i \mapsto d? \ i \} \\
\hline
\end{array}
$$

Another example

$$
\begin{array}{|l}
\hline
DetectReferenceEvent_{18}^{NR} \\
now, fa : \mathbb{N} \\
ot : REVENT \pfun \mathbb{N} \\
tli, tls, X : REVENT \to \mathbb{N} \\
sysState : STATUS \\
e? : REVENT \\
\hline
e? = LO \\
sysState = normal \\
e? \notin \operatorname{dom} ot \\
now \in tli\ e? \upto tls\ e? \\
X\ e? \leq fa \\
ot \neq \emptyset \\
\{e? \mapsto now\} \neq \emptyset \\
ot \cap \{e? \mapsto now\} = \emptyset \\
1 < now < 3 \\
\hline
\end{array}
$$

Test cases in the TTF

A test case is an element of a test template. For example:

$$
\begin{array}{l}
DetectReferenceEvent_{18}^{TC} \\
DetectReferenceEvent_{18}^{NR} \\
\hline
tli = \{LO \mapsto 2, TD1E \mapsto 5, TD2E \mapsto 4, TD3E \mapsto 10\} \\
tls = \{LO \mapsto 10, TD1E \mapsto 12, TD2E \mapsto 14, TD3E \mapsto 16\} \\
X = \{LO \mapsto 3, TD1E \mapsto 5, TD2E \mapsto 7, TD3E \mapsto 9\} \\
e? = LO \\
sysState = normal \\
now = 2 \\
fa = 10 \\
ot = \{TD1E \mapsto 3\}
\end{array}
$$

That is, a test case is an element satisfying a first-order predicate over the Zermelo-Fraenkel set theory.

Finding Test Cases in the TTF

- As noted by Cavalcanti and Gaudel, finding a test case in the TTF can be very difficult.
- Currently, Fastest implements a rough satisfiability algorithm:
  - Finite sets are taken for variables and types in a test template.
  - The predicate is evaluated on each element of the Cartesian product of those finite sets.
    - With the ZLive component of the CZT project.
  - Some simple heuristics are applied.
  - The mathematical structure of the predicate is not considered.
- Fastest has been able to find test cases for an average of 80 % of the satisfiable test templates from eleven case studies.
  - Some of these case studies are real problems.
- Nevertheless, Fastest's algorithm needs to be improved or replaced by more sophisticated ones.

SMT solvers for the TTF

Satisfiability Modulo Theory (SMT) solvers

SMT solvers are tools that solve the problem of satisfiability for a range of mathematical and logical theories.

- No SMT solver implements the Zermelo-Fraenkel set theory.
- Must define a shallow embedding.
- Many questions:
  - Is the language of SMT solvers expressive enough?
  - Is there just one shallow embedding?
  - Will the chosen embedding solve all the satisfiable test specifications appearing in the TTF and real Z specifications?
  - Which SMT solver and which embedding will be the best in satisfying more test specifications in less time?
  - Will it be better than Fastest in finding test cases?
  - Or should the SMT solver complement Fastest in this task?

Our contribution

Two shallow embeddings

During this work we defined a shallow embedding of the Zermelo-Franekel set theory for two mainstream SMT solvers: Yices and CVC3.

Initial empirical assessment

We manually translated 69 satisfiable test templates, for which Fastest cannot find a test case, into these shallow embeddings and ran the SMT solvers over them.

In this talk we will only show:

- Some details of the shallow embedding for Yices.
- The results of the empirical assessment.

$$\frac{A : \mathbb{P}\, X}{A : X \to \mathbb{B}} \qquad \frac{R : X \leftrightarrow Y}{R : X \times Y \to \mathbb{B}}$$

So far, so good ...

$$\frac{\emptyset : X}{\mathit{emptyset}X : (X \to \mathbb{B})(\lambda\ \ (x : X)\ \mathsf{false})}$$

First problem:

- Yices does not support polymorphic operators so we need one empty set per type.
- This is not really serious since we intend the shallow embedding to be hidden from Fastest's users.

$$\frac{f : X \pfun Y}{f : [dom : X \to \mathbb{B}, law : X \to Y]}$$

More problems:

- In Z partial functions are sets. That is:
  - They can be applied: $f\ x$.
  - But it is also legal and useful to use them as sets: $f \cup g$.
- The embedding does not preserve this notion.
  - The set underlying a partial function can be calculated:

$$fSet : (X \times Y \to \mathbb{B})$$
$$(\lambda\ (x : X;\ y : Y)\ f.dom\ x \wedge f.law\ x = y)$$

- Net balance: it works (other attempts are worse).

$$\frac{A : \mathbb{F}\, X}{A : [set : X \to \mathbb{B}, bij : X \to \mathbb{N}_1, card : \mathbb{N}]}$$

and the following axioms:

$$\forall\, x : X \bullet A.set\ x \Leftrightarrow A.bij\ x \leq A.card$$

$$\forall\, n : \mathbb{N}_1;\ x_1, x_2 : X \bullet$$
$$(n \leq A.card \land A.set\ x_1 \land A.set\ x_2 \land A.bij\ x_1 = A.bij\ x_2 = n)$$
$$\Rightarrow x_1 = x_2$$

Even more problems:

- According to the embedding, finite sets and sets are quite different, while in Z they are almost equal.
- Universal quantifications!

- We use eleven case studies (Z specifications) to test and validate Fastest.
  - Fastest finds 100 % of the test cases for two of them.
  - One was discarded because its test templates are too complex.
- In the remaining eight case studies, there are 69 satisfiable test templates for which Fastest fails to find a test case.
- As a first step in this empirical assessment:
  - These 69 test templates were manually translated into the shallow embeddings.
  - We also tried a variant of the embeddings where basic types were replaced by enumerated types with three elements.

$$[X] \qquad \longrightarrow \qquad X ::= x_1 \mid x_2 \mid x_3$$

  - The resulting 69 formulas were provided to the SMT solvers.
  - How many witnesses were found?

Empirical assessment
Results

Case study	Main embeddings				Variant embeddings			
	Yices		CVC3		Yices		CVC3	
	Sat	**Unk**	**Sat**	**Unk**	**Sat**	**Unk**	**Sat**	**Unk**
Savings accounts (3)	8		8		8		8	
Savings accounts (1)		2		2		2		2
Launching vehicle		8	8			8	8	
Plavis		29		29		29		29
SWPDC		13		13		13		13
Scheduler		4		4		4		4
Security class		4		4		4		4
Pool of sensors	1		1		1		1	
Totals	**9**	60	**17**	52	9	60	17	52

Time spent in processing all the 69 test templates:

- Yices: 3 s
- CVC3: 420 s
- Fastest: 390 s (but no test case is discovered)

- CVC3 discovered the same test cases than Yices, plus 8 more.
- However, Yices ran faster.
- The variant embedding produced the same results.
- Yices could not solve all the test templates including a quantification or a lambda expression over an infinite set.
- CVC3 could not solve all the test templates including a quantification over an infinite set.

These quantifications or lambda expressions are not present in the original Z test templates!

Conclusions

- Given that there are 475 satisfiable test templates in the 8 case studies, CVC3 would add only 4 % of test cases.
  - Assuming CVC3 finds all the test cases that Fastest does.
  - Why such a powerful tool would perform barely better than our silly algorithm?
- What about potential witnesses?
  - At first glance many of them seem good candidates.
  - Use ZLive to confirm.
- How easy is to translate the witnesses back to Z?
  - Yices produced 1,000 lines of output.
  - CVC3 produced 32,000!

Future work

- What happens with the other satisfiable test templates?
- Z3.
- A decision procedure for Z predicates?