The Great Book of Zebra

The Zebra Project

September 11, 2015

Preface

This book is a collaborative work from the https://github.com/fsvieira/zebrajs project community and everyone is invited to participate.

The list of contributors is at the contributors section 1.3 and your name can be there too :D.

This is a work in progress.

Contents

Chapter 1

Introduction

This is the official book of Zebra-machine (ZM). Here you will find anything you need to understand in deep the ZM, the book covers both theoretical and practical definitions.

Zebra-machine (ZM)is a logical symbolic computation query system, given a set of computational definitions it will answer questions about them, therefor ZMis better suited for software validation and constrain satisfaction problems.

1.1 The Zebra-machine (ZM)

As mentioned before ZM is a logical symbolic computation query system, and it consists of two parts the definitions and the query, both parts share the same language of ZM terms, which is defined by a certain formal syntax, and a set of transformation rules.

1.1.1 ZM Language ($\mathbb{L}$)

The ZM language ($\mathbb{L}$) of terms are defined as:

1. $c \in \mathbb{C}$: $\mathbb{C}$ is the set of terminal symbols called constants, c is a terminal symbol. Constants are ZM terms.

2. $'p \in \mathbb{V}$: $\mathbb{V}$ is the set of variables, p is a variable. Variables are ZM terms.

3. $(p_0 \ldots p_n) \in \mathbb{T}$: $\mathbb{T}$ is the set of tuples, $(p_0 \ldots p_n)$ its a n-tuple of ZM terms and is a ZM term.

4. $\sigma \to (p_0 \ldots p_n) \in \mathbb{T} \wedge \sigma \in \mathcal{P}(\mathbb{V})$

5. $p \otimes q$

6. $p \ominus q$

7. Nothing else is a ZM term.

1.1.2 ZM Operations

Unification ($\otimes$, binary operation) defined as

$$\otimes : \mathbb{L} \times \mathbb{L} \to \mathbb{L}$$

and by the rules,

1. $p \otimes p \implies p$

   p unifies with itself, resulting on itself.

2. $'p \otimes Q \iff {'p} = Q,\ p \in \mathbb{V} \wedge Q \in \mathbb{L}$

   $'p$ is a variable and Q is a ZM term, they unify iff $'p = Q$.

3. $Q \otimes' p \iff {'p} = Q,\ p \in \mathbb{V} \wedge Q \in \mathbb{L}$

   $'p$ is a variable and Q is a ZM term, they unify iff $'p = Q$.

4. $(p_0 \dots p_n) \otimes (q_0 \dots q_n) \iff (p_0 \otimes q_0 \dots p_n \otimes q_n)$

   $(p_0 \dots p_n)$ z-tuple only unifies with other z-tuple if they have same size and all sub ZM terms unify.

5. Anything else fails to unify.

Not-unify ($\ominus$, binary operation) defined as

$$\ominus : \mathbb{L} \times \mathbb{L} \to \mathbb{L}$$

and by the rules,

1. $P \ominus Q = P \iff \overline{P \otimes Q},\ P \in \mathbb{L} \wedge Q \in \mathbb{L}$

   Two ZM terms not-unify if they dont unify.

2. Note:

   In case of variables their values must also not-unify,

   Tuples and constants will never unify,

   If two tuples are not-unifiable then at least one of the elements is not-unifiable.

Substitution ($\mathcal{S}$, function) defined as

$$\mathcal{S} : \mathbb{V} \times \mathbb{L} \times \mathbb{L} \to \mathbb{L} \tag{1.1}$$

$$\mathcal{S}(v, w, t) = \begin{cases} w & \text{, if } t = v, \\ (\mathcal{S}(v, w, t_0) \dots \mathcal{S}(v, w, t_n)) & \text{, if } t = (t_0 \dots t_n) \\ t & \text{, otherwise} \end{cases} \tag{1.2}$$

1.1.3 ZM Computation

A ZM computation is expressed as 4-tuple $(\sigma, \delta, q, \alpha)$ **where:**

1. σ is a set of terminal symbols (constants),
2. δ is a set of z-tuples (definitions),
3. q is a z-tuple (query),
4. α is the set of possible computational answers to query q based on *delta* definitions.

A definition is a fact in the system. The inner tuples of a definition are considered and called queries, therefor for a definition to be true all of its inner tuples/queries must also be true.

A query is a question to the system that is true if and only if it unifies at least with one definition.

Free and bound variables on the context of a definition all definition variables are considered to be bound to the definition, on the context of queries all variables are free.

1.2 Computing Examples

Unification

1. $yellow \otimes yellow \implies yellow$

   succed.

2. $blue \otimes yellow$

   fail: can't unify constants with diferent value.

3. $yellow \otimes (yellow)$

   fail: can't unify constant and tuple.

4. $(blue\ yellow) \otimes (blue\ yellow) \implies (blue \otimes blue\ yellow \otimes yellow) \implies (blue\ yellow)$

   succed.

Not-Unify

- $yellow \ominus blue$,

  $yellow$ and $blue$ are constants and $yellow \neq blue$.

- $(blue\ yellow) \ominus (yellow\ blue)$,

  $(blue\ yellow) \neq (yellow\ blue)$

- $(blue\ 'p) \ominus (yellow\ blue)$,

  $'p$ is a variable and since $(blue'p) \neq (yellow\ blue)$ then $'p \neq blue$

- $'p \ominus' q$,

  $'p$ and $'q$ are variables, $'p \neq' q$.

1.3 Contributors

- Filipe Vieira, https://github.com/fsvieira