Representing Expressive Types In Blocks Programming Languages

Marie Vasek, Wellesley College
NEPLS
June 1, 2012
**Representing types from functional languages in blocks languages**

**Problem:** Current block languages aim to lower barriers to programming but only make weak attempts to implement static types and do not represent tree-structured types.

**Solution:** Create a blocks language where the shape of the block connector reflects the tree structure of a type.

**Overview:**
- Type systems in other blocks languages
- **TypeBlocks:**
  - Shape types
  - Polymorphism
  - Work in progress: functions, algebraic data types
- All types have the same connector shape
- Most type checking at run time, but some at connection time
TIMING OF ERRORS

This block cannot plug into this socket because hello is not a number.

The Troubleshooting Guide at http://appinventor.googlelabs.com/learn/troubleshooting.html may provide further help.

Bad arguments to log

The operation log cannot accept the arguments: hello24

End Application
SCRATCH – WOOLY TYPING

- Three primitive types (boolean, string, number)
- Three shapes (angle = boolean, rounded = string or number, box = any)
**Type Conversion**

Evaluate to true

Evaluates to false
**TYPE CONVERSION - LISTS**

- **xs**
  - 1: 1
  - Length: 1

- **ys**
  - 1: true
  - Length: 1

- **xs**
  - 1: 37
  - Length: 2

- **ys**
  - 1: 37 foo
  - Length: 1

- **false**

- **true**
BYOB – More Wonkiness
StarLogo: TNG

6 types:
POLYMORPHISM
Buggy Procedure Typing
WHAT I DID

- Blocks types inspired by SML
- Base types + type constructors
  - => ability to represent countably many types
- Each arbitrarily complex type = unique connector shape
- ML-style universal polymorphism
- … but no blocks language constructed from this yet
  - no functions or algebraic datatypes
Base Types

3 base types: number, boolean, string
**Build-a-Type**

3 constructors:

- listof
- pair
- function

| listof int | listof (listof string) | int * string | bool -> string |
MORE EXAMPLE PLUGS

listof (string * boolean)  (listof string) * boolean

boolean  * (string -> listof number)
Pop Quiz
Zip and Map

mapZip
over
TYPES TO SHAPES

- Recursive drawing method
- Draw:
  - Bottom of arrow
  - Range argument
  - Middle of the arrow
  - Domain argument
  - Top of the arrow

- Smallest type has size unit
- 2 arguments: take the max
TYPE CONSTRUCTION IN PRACTICE

DEMONSTRATION
POLYMORPHISM

On events connection and disconnection

On Connection:
- Unifies types of socket and plug
- If type of plug / socket changes:
  - Propogate change to all uses of that polymorphic type

On Disconnection:
- “Reset” type
- Propogate type changes to the parent / children
IMPLEMENTATION DETAILS

- ScriptBlocks
- in JavaScript using Google Closure Library
- Represent recursive types by strings and objects

```javascript
{"funD": {"tupX": "boolean", "tupY": "string"},
  "funR": {"listOf": "number"}}
```

- Represent poly types by objects
  - Ie `{“poly” : “a”}` or `{“poly”: “b”}` where “a” and “b” are like sml’s ‘a and ‘b.
What should functions look like?

StarLogo: TNG

BYOB
What should algebraic datatypes look like?

And

Or
FOR A LATER DATE

- A sml-like statically typed functional blocks language using these types
  - differentiating visually between ‘a and ‘b.
    - better visualization of polymorphic types
  - algebraic data types
  - pattern matching

- Usability
  - highlighting of all compatible connections
  - user testing

- Other languages and static semantics features
  - Object typing
  - Exception propagation
  - effects

- Other representations of type
  - Waterbear – types as color
  - any others???
WATERBEAR
- Inspired by Scratch
- Represents type through color
- 4 basic types: boolean, number, string, array + “all” type
- Explicit casting to convert types
IDEAS FOR COMPOSABLE TYPES - COLOR
any questions?

polyCons to []

any questions?