Towards improved abstractions for programming language processor specifications

Shirley Goldrei
Programming Languages Research Group
Macquarie University

SAPLING 11th November 2006

Motivation

- Premise: Domain Specific Languages
  - improve productivity
  - reduce programming errors
  - leverage technology improvements
- Ideal goal: Make it possible for domain experts to specify domain specific languages without needing to be compiler experts
- Strategy: Generate compilers from high level specifications

Context

- Specifying language semantics
- Specifying semantics preserving transformations
- Compilation is an example of semantics preserving transformations (see GHC Haskell compiler)
- Abstract Syntax (input and output)
  (i.e. assumes a predefined parsing and unparsing)

Ways of Specifying Language Semantics

Attribute Grammar

- Syntax Driven
- Specify attribute dependencies
- Tree walks are inferred
- Output trees are constructed

Term Rewriting

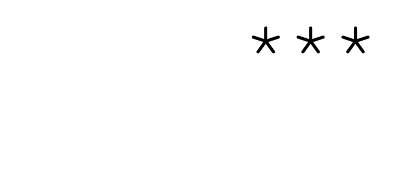
- Syntax Driven
- Tree walks are either fixed or explicitly defined
- Input tree is transformed into output tree in place

Two little (domain specific) languages…

- Single domain
- Context sensitive semantics
- Non-trivial transformation
- Require computation to preserve semantics
- Multiple passes required

SPLD1

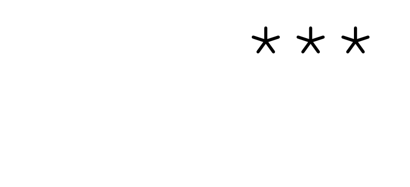
```
PEN UP
  DRAW 5
NEP
PEN DOWN
  DRAW 3
  NEWLINE
NEP
```



```
     ***
```

SPLD2

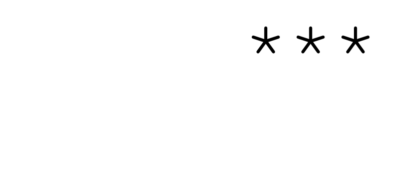
```
5 1 MOVETO
3 DRAW
```



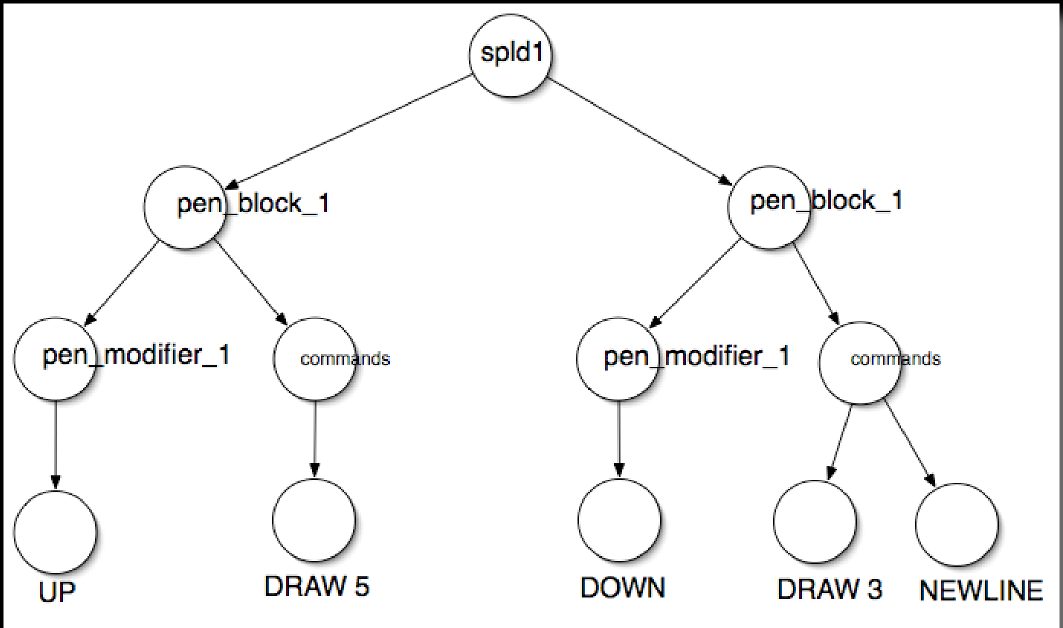
```
     ***
```

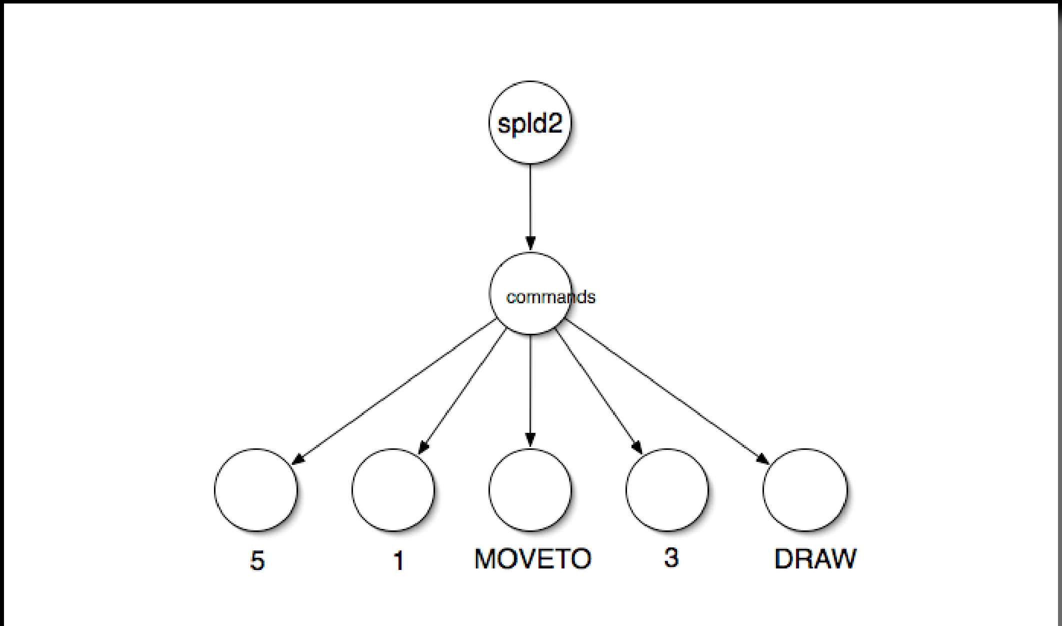
SPLD2

```
3
5
1
MOVETO
DRAW
```



```
***
```


spld1
pen_block_1
pen_block_1
pen_modifier_1
commands
pen_modifier_1
commands
UP
DRAW 5
DOWN
DRAW 3
NEWLINE


spld2
commands
5
1
MOVETO
3
DRAW

Research Questions

- Can we devise an improved abstraction?
  - Incorporate strengths of existing systems
    - Inferred traversals
    - “conceptually” in-place rewrites
  - Improve expressivity of semantic preservation
    - explicit notions of *input* and *output* grammars
    - “Type safety” (where required) in terms of input and output grammars
- Oh and while we are at it… Can we deal with graphs and not just trees, without adding too much cognitive overload?

Research Questions

- Relationship between transformation and mechanism features?
  - Along what dimensions can we measure a *syntactic or semantic gap* between languages?
  - What can we say about how “similar” or “different” languages are? How similar are say, Pascal and C? What about C++ and Java?
  - Can we formalise our intuition? (e.g. develop a partial order or measure)
  - How does the similarity of the languages relate to the usefulness of the transformation tools
- Understanding the nature of the semantic gap will help to inform the development of an improved abstraction

Experiments

- Familarisation: Implement translations in a number of different systems (e.g. TXL, Stratego/XT, UUAG, JastAdd, Eli)
- Understand performance implications: Translate several thousand line real world application

Plan

- Develop a framework for analysing programming language translation tasks
- Comparing alternate systems against the framework
- Develop an improved abstraction for programming language processor specification

Thank You

Comments and questions welcomed

[plrg] macquarie

MACQUARIE
UNIVERSITY~SYDNEY