# Assignment Program Representation

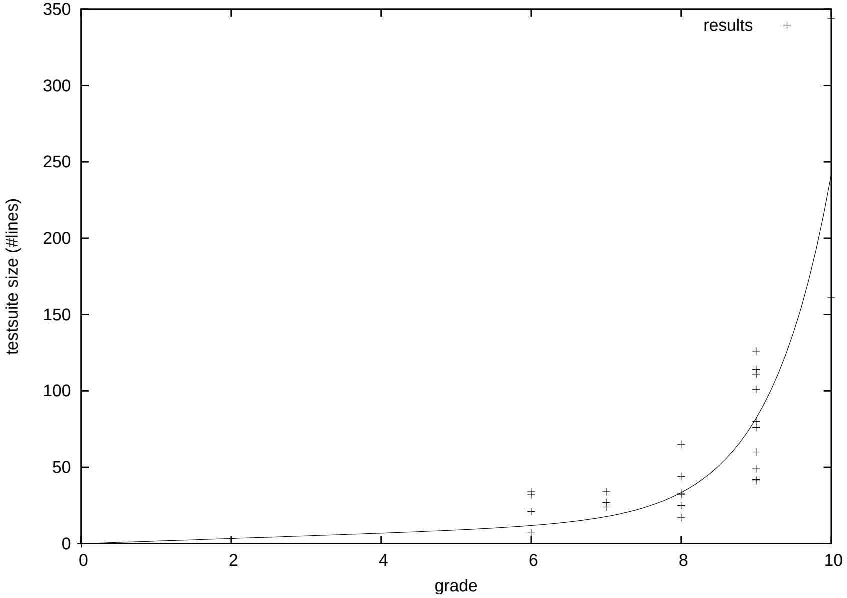
Master Course Program Transformation 2005-2006

Martin Bravenboer

Institute of Information & Computing Sciences
Utrecht University,
The Netherlands

February 21, 2006

# Testing Pays off . . .



## Constructors

```
regular tree grammar
  productions
    Form -> False()
    Form -> True()
```

```
context-free syntax
  "true"  -> Form {cons("True")}
  "false" -> Form {cons("False")}
```

```
$  echo "true" | sglri -p Example.tbl
True()
```

```
lexical syntax
  "true" -> True
  "false" -> False
context-free syntax
  True  -> Form {cons("True")}
  False -> Form {cons("False")}
```

```
$  echo "true" | sglri -p Example.tbl
True("true")
```

```
lexical syntax
  "true"  -> BoolConst
  "false" -> BoolConst
context-free syntax
  BoolConst  -> Form {cons("Bool")}
```

```
$  echo "true" | sglri -p Example.tbl
Bool("true")
```

## Priority of And/Or

### And binds stronger than Or

```
    > Form "/\\" Form -> Form
    > Form "\\/" Form -> Form
```

### Wrong: And and Or not in the same priority group.

```
  > {left:
      Form "\\/" Form -> Form
      Form "/\\" Form -> Form
    }
```

### Wrong: And and Or are usually not assoc

```
  > { Form "/\\" Form -> Form
      Form "\\/" Form -> Form
    }
```

### Wrong: Group with single production is not useful

```
  {left:
    Form "/\\" Form -> Form
  }
```

## Priorities

### Wrong: No reason for Pred and Id in priorities

```
context-free priorities
  { Id        -> Form
    "true"    -> Form
    "false"   -> Form }
> {
    Id "(" {Form ","}* ")" -> Form
    "not" Form     -> Form }
> ...
```

### Wrong: No assoc group required for single production

```
context-free syntax
  Form "/\\" Form -> Form {left}
context-free priorities
  {left:
    Form "/\\" Form -> Form
  }
```

### Wrong: No curly braces for single production

```
context-free priorities
  > { Form "/\\" Form -> Form }
  > ...
```

# Associativity of Exists and Forall

## Exists and Forall should be in same group

```
context-free priorities
    ...
  > { "forall" Id ":" Form -> Form
      "exists" Id ":" Form -> Form }
```

## Optional, right assoc. Not left!

```
context-free priorities
    ...
  > {right:
      "forall" Id ":" Form -> Form
      "exists" Id ":" Form -> Form
    }
```

## Wrong: Forbids `exists` as child of forall

```
context-free priorities
    ...
  > "forall" Id ":" Form -> Form
  > "exists" Id ":" Form -> Form
  > ...
```

## Not, Exists and Forall

**‘Wrong’: forbids forall and exists as argument of not**

```
context-free priorities
    "not" Form -> Form
  > ...
  > {
      "forall" Id ":" Form -> Form
      "exists" Id ":" Form -> Form
    }
```

Bummer: Cannot be solved easily!

## Keywords and Identifiers

Reject keywords as identifiers

- Not a problem in most solutions
- However, use non-terminal `Keyword`.

Follow restriction for keywords

- All keywords, not just `true` and `false` (`notx`)
- Does not work: `KeyWord -/- [A-Za-z0-9]`

Follow restriction on Id

- `Id -/- [A-Za-z0-9]`
- Useful, but not necessary for PLF

## Follow Restriction for Layout

`Pred( )` is ambiguous

- Space can be before or after the empty list
- Solution: follow restriction on optional layout

```
context-free restriction
  LAYOUT? -/- [\ \t\n\r]
```

Not a solution:

```
lexical restriction
  LAYOUT -/- [\ \t\n\r]
```