Centrum Wiskunde & Informatica

RLSRunner and KRunner: Linking Rascal with K for Program Analysis and Execution

Mark Hills, Paul Klint, & Jurgen J. Vinju

2nd International Workshop on the K Framework and its Applications
August 9, 2011
Cheile Gradistei, Romania

k-framework
The K Semantic Framework

Overview

Overview

- Motivation

Overview

- Motivation
- Tool components: Rascal

Overview

- Motivation
- Tool components: Rascal
- Tool components: K

Overview

- Motivation
- Tool components: Rascal
- Tool components: K
- Demo

Overview

- Motivation
- Tool components: Rascal
- Tool components: K
- Demo
- Wrap-up

Motivation: Why This Tool?

Motivation: Why This Tool?

- Many K and K in Maude specifications exist -- want reuse

Motivation: Why This Tool?

- Many K and K in Maude specifications exist -- want reuse
- Integrating with graphical environments currently ad-hoc, bad user experience

Motivation: Why This Tool?

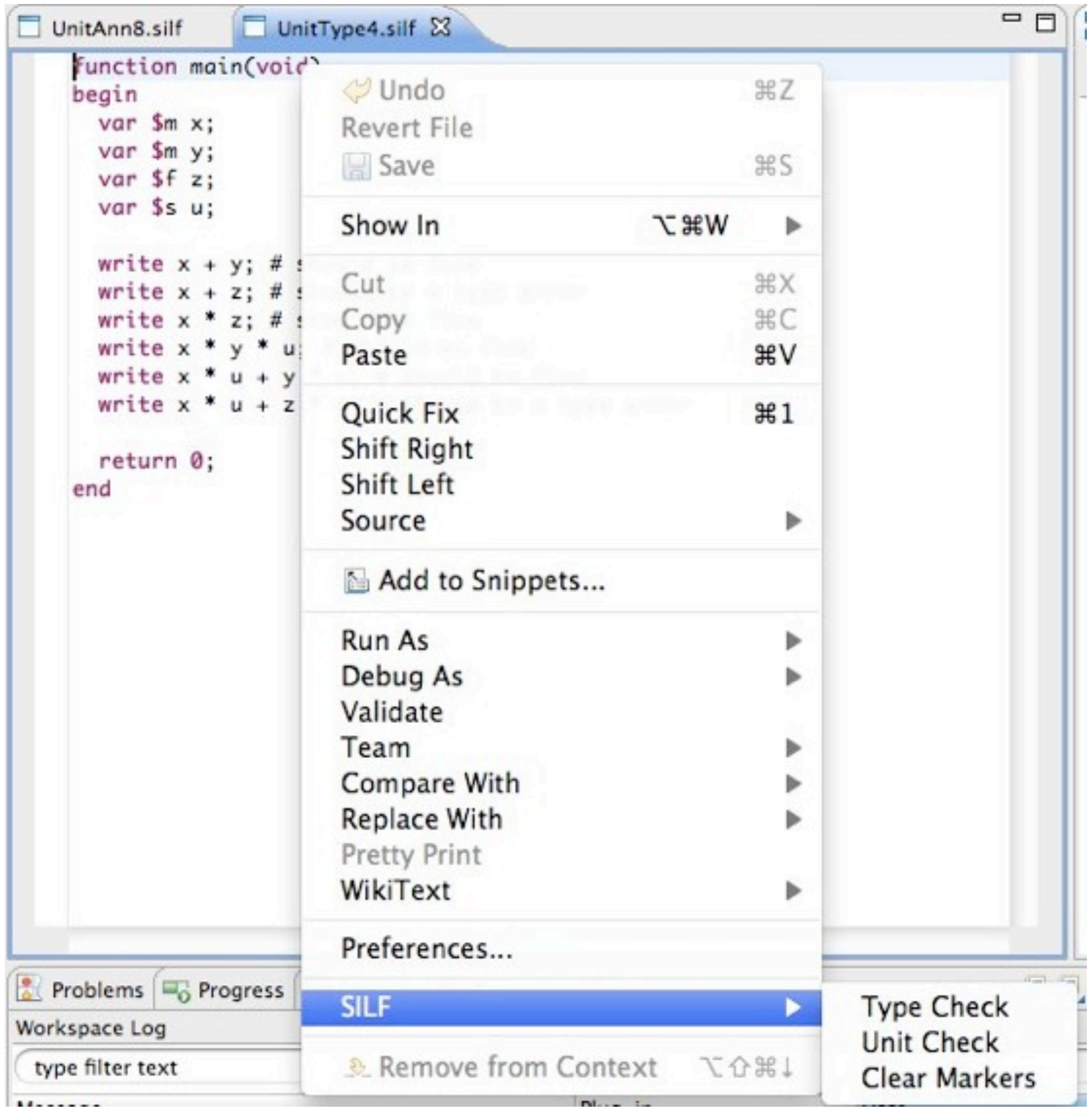
- Many K and K in Maude specifications exist -- want reuse
- Integrating with graphical environments currently ad-hoc, bad user experience
- Want a general method to integrate these specifications with Rascal-based IDEs

Motivation: Why This Tool?

- Many K and K in Maude specifications exist -- want reuse
- Integrating with graphical environments currently ad-hoc, bad user experience
- Want a general method to integrate these specifications with Rascal-based IDEs
- (Personal) Wanted something like this all during my PhD

How Should it Work (RLSRunner)?

How Should it Work (RLSRunner)?



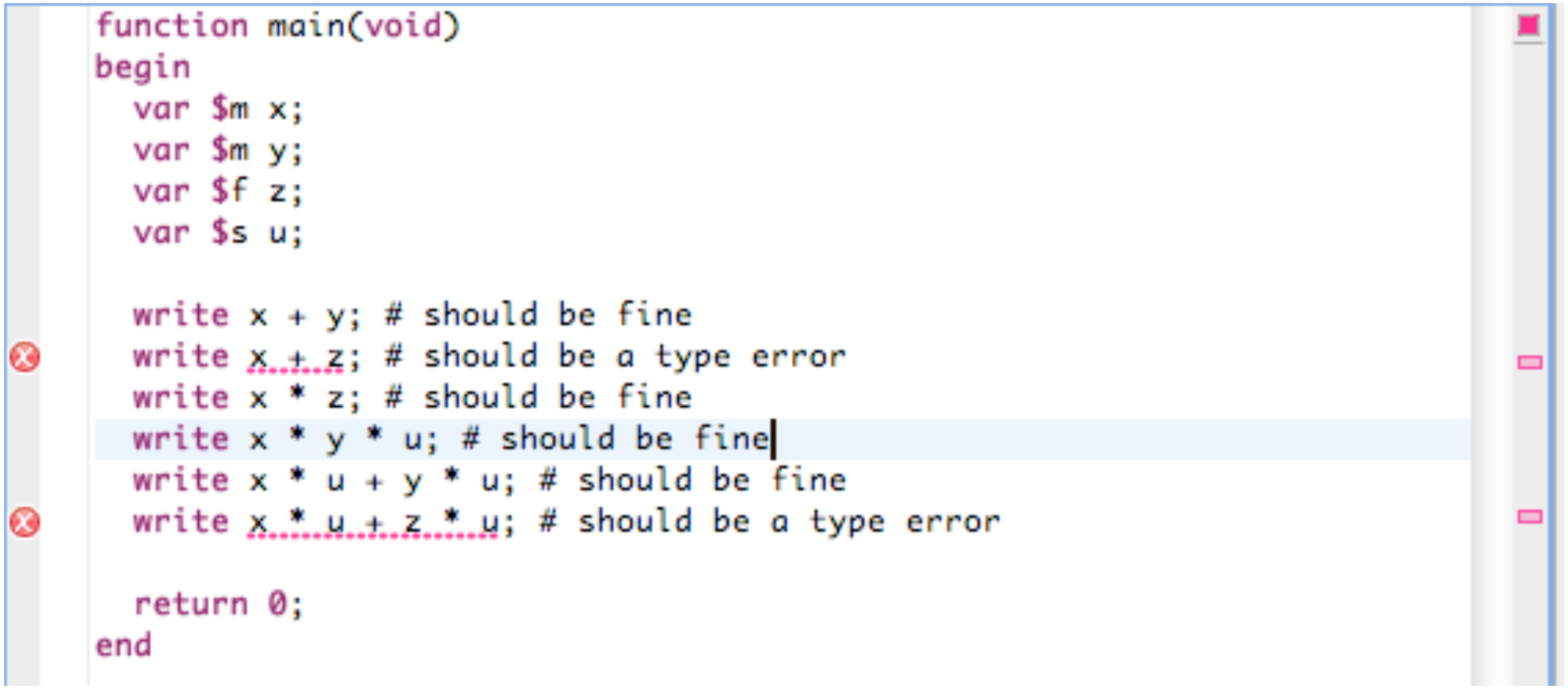
```
function main(void)
begin
  var $m x;
  var $m y;
  var $f z;
  var $s u;

  write x + y; # s
  write x + z; # s
  write x * z; # s
  write x * y * u;
  write x * u + y
  write x * u + z

  return 0;
end
```

How Should it Work (RLSRunner)?

How Should it Work (RLSRunner)?



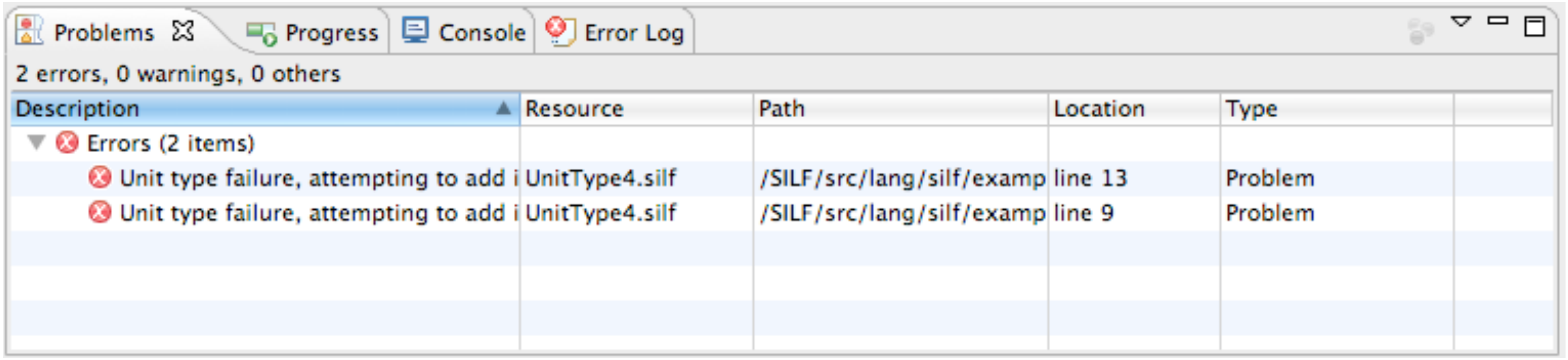
```
function main(void)
begin
  var $m x;
  var $m y;
  var $f z;
  var $s u;

  write x + y; # should be fine
  write x + z; # should be a type error
  write x * z; # should be fine
  write x * y * u; # should be fine
  write x * u + y * u; # should be fine
  write x * u + z * u; # should be a type error

  return 0;
end
```

How Should it Work (RLSRunner)?

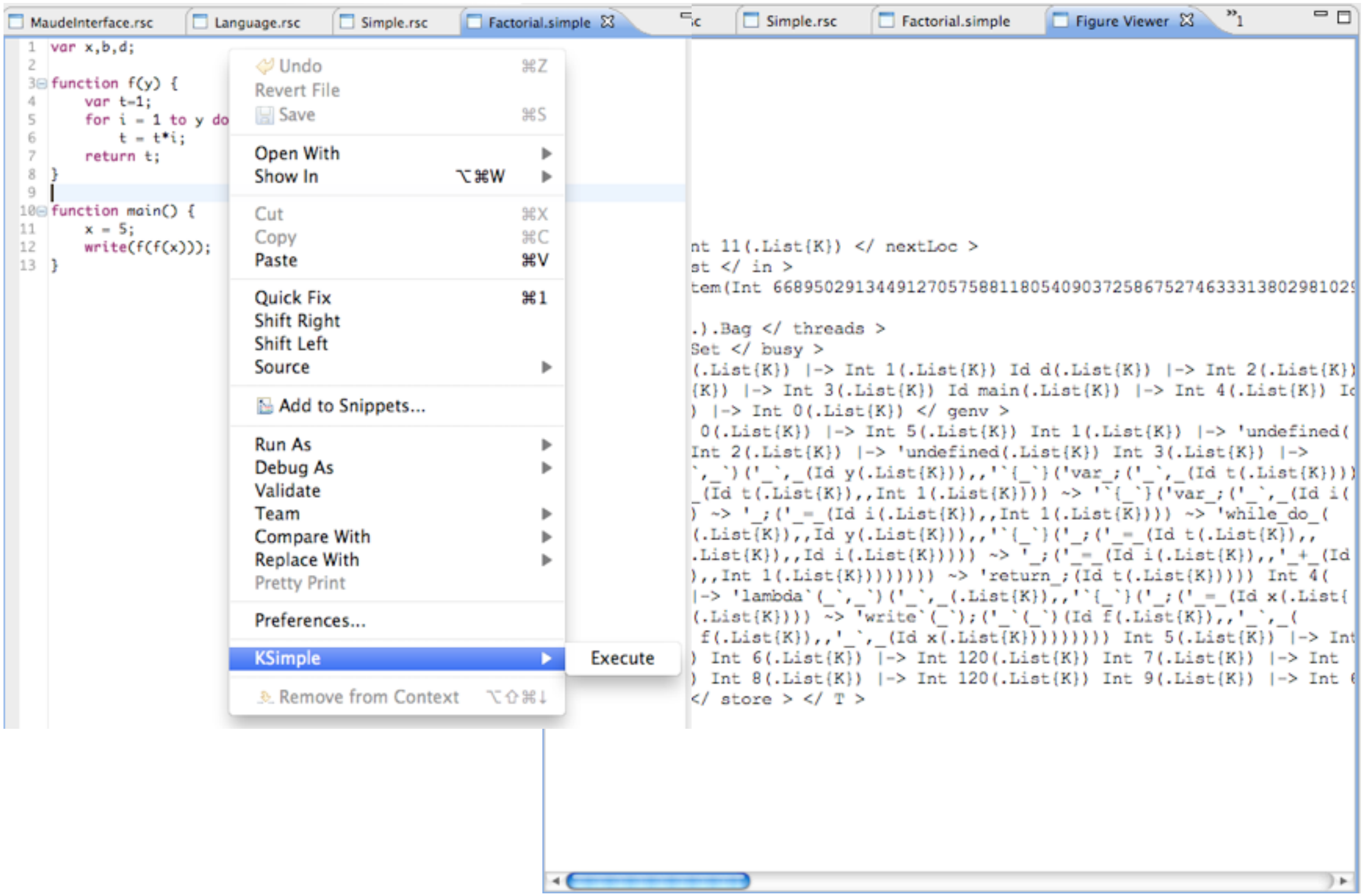
How Should it Work (RLSRunner)?



2 errors, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (2 items)				
Unit type failure, attempting to add i	UnitType4.silf	/SILF/src/lang/silf/examp	line 13	Problem
Unit type failure, attempting to add i	UnitType4.silf	/SILF/src/lang/silf/examp	line 9	Problem

How Should it Work (KRunner)?



```
var x,b,d;

function f(y) {
    var t=1;
    for i = 1 to y do
        t = t*i;
    return t;
}

function main() {
    x = 5;
    write(f(f(x)));
}
```

```
nt 11(.List{K}) </ nextLoc >
st </ in >
tem(Int 66895029134491270575881180540903725867527463331380298102

.).Bag </ threads >
Set </ busy >
(.List{K}) |-> Int 1(.List{K}) Id d(.List{K}) |-> Int 2(.List{K})
{K}) |-> Int 3(.List{K}) Id main(.List{K}) |-> Int 4(.List{K}) I
) |-> Int 0(.List{K}) </ genv >
 0(.List{K}) |-> Int 5(.List{K}) Int 1(.List{K}) |-> 'undefined(
Int 2(.List{K}) |-> 'undefined(.List{K}) Int 3(.List{K}) |->
`,_`)('_`,_(Id y(.List{K})),,'`{_`}('var_;('_`,_(Id t(.List{K})))
_(Id t(.List{K}),,Int 1(.List{K}))) ~> '`{_`}('var_;('_`,_(Id i(
) ~> '_;('_=_(Id i(.List{K}),,Int 1(.List{K}))) ~> 'while_do_(
(.List{K}),,Id y(.List{K})),,'`{_`}('_;('_=_(Id t(.List{K}),,
.List{K}),,Id i(.List{K})))) ~> '_;('_=_(Id i(.List{K}),,'_+_(Id
),,Int 1(.List{K}))))))) ~> 'return_;(Id t(.List{K})))) Int 4(
|-> 'lambda`(_`,_`)('_`,_(.List{K}),,'`{_`}('_;('_=_(Id x(.List{
(.List{K}))) ~> 'write`(_`);('_`(_`)(Id f(.List{K}),,'_`,_(
 f(.List{K}),,'_`,_(Id x(.List{K}))))))) Int 5(.List{K}) |-> Int
) Int 6(.List{K}) |-> Int 120(.List{K}) Int 7(.List{K}) |-> Int
) Int 8(.List{K}) |-> Int 120(.List{K}) Int 9(.List{K}) |-> Int
</ store > </ T >
```

An Introduction to Rascal

- Rascal: A meta-programming language for source code analysis and transformation

- Based on concepts learned from ASF+SDF, but with a more traditional programming language feel

- Features: parsing, structured control flow, rich data types (algebraic data types, lists, sets, tuples, maps, relations, etc), pattern matching, enumerations, higher order functions, etc

Defining Grammars in Rascal

Tool Components: Rascal (ShellExec)

Rascal

```
PID pid = createProcess(maudeLocation.path);
writeTo(pid, toRun);
res = readFrom(pid);
killProcess(pid);
```

Tool Components: Rascal (ResourceMarkers)

Rascal

```
data Message = error(str msg, loc at)
             | warning(str msg, loc at)
             | info(str msg, loc at);
```

Rascal

```
import Message;

public void java removeMessageMarkers(loc resourceLoc);

public void java addMessageMarkers(set[Message] markers);
```

Tool Components: Rascal (RLSRunner ADT)

Rascal

```
data RLSRunner = RLSRun(loc maudeFile,
                        str(str,list[str]) pre,
                        RLSResult(str) post);
```

Tool Components: Rascal (Maude-ifier)

Rascal

```
if ((Program)`<Decl* decls> <FunDecl+ funDecls>` := p)
   return located(p,"Pgm",
            "__(<showDecls([d|d<-decls])>,
                <showFunDecls({f|f<-funDecls})>)");
```

Rascal

```
syntax Program = Default: Decl* decls FunDecl+ funDecls;
```

Tool Components: Rascal (Returning Results)

Rascal

```
data RLSResult = SILFAnalysisResult(bool foundErrors,
                                    set[Message] messages) ;
```

Rascal

```
void exec(Tree pt, loc l) {
  str pgm = maudeify(pt, true, policy);
  RLSRunner rlsRunner = RLSRun(silfSpec, pre, post);
  RLSResult res = runRLSTask(maudeExec, rlsRunner, pgm);
  if (SILFAnalysisResult(true,msgs) := res)
    addMessageMarkers(msgs);
}
```

Tool Components: Rascal (Generate Program Files)

Rascal

```
public str generateProgramModule(Tree pgm, str topSort, str pgmName,
                                 str pgmMod, str syntaxMod) {
    set[str] identifiers = { "<id>" | /Id id <- pgm } - "main";
    str identifierListing =
       "syntax Id ::= <intercalate(" | ", [i|i<-identifiers])> ";
    str pgmDeclaration = "syntax <topSort> ::= <pgmName>";
    return "kmod <pgmMod> is including <syntaxMod>
           '<identifierListing>
           '<pgmDeclaration>
           '
           'macro <pgmName> =
           '  <pgm>
           '
           'endkm
           '";
}
```

Tool Components: K (Rascal Source Locations)

K/Maude

```
fmod RASCAL-LOCATION is
  including STRING .
  including INT .
  sort RLocation .
  op sl : String Int Int Int Int Int Int -> RLocation .
endfm
```

Tool Components: K (Location Semantics)

K/Maude

```
op currLoc : RLocation -> State [format (r! o)] .

op rloc : RLocation -> ComputationItem .

eq k(rloc(RL) -> K) currLoc(RL') = k(K) currLoc(RL) .

eq k(exp(locatedExp(E, RL)) -> K) currLoc(RL') =
    k(exp(E) -> rloc(RL') -> K) currLoc(RL) .
```

Tool Components: K (Generating Results)

K/Maude

```
op makeAnalysisMsg : OutputList -> String .

eq makeAnalysisMsg(warning(level(1) msgloc(RL) msg(S) WIS), OL) =
    ("||1:::" + rloc2str(RL) + ":::" + S + "||") + makeAnalysisMsg(OL) .
```