Extensible Compiler Architecture Examples from JModelica.org

Uses of the JastAdd systems

Görel Hedin
Computer Science, Lund University

LCCC workshop, Lund, Sept 20, 2012

Background

JastAdd: an open source metacompiler for generating extensible compilers

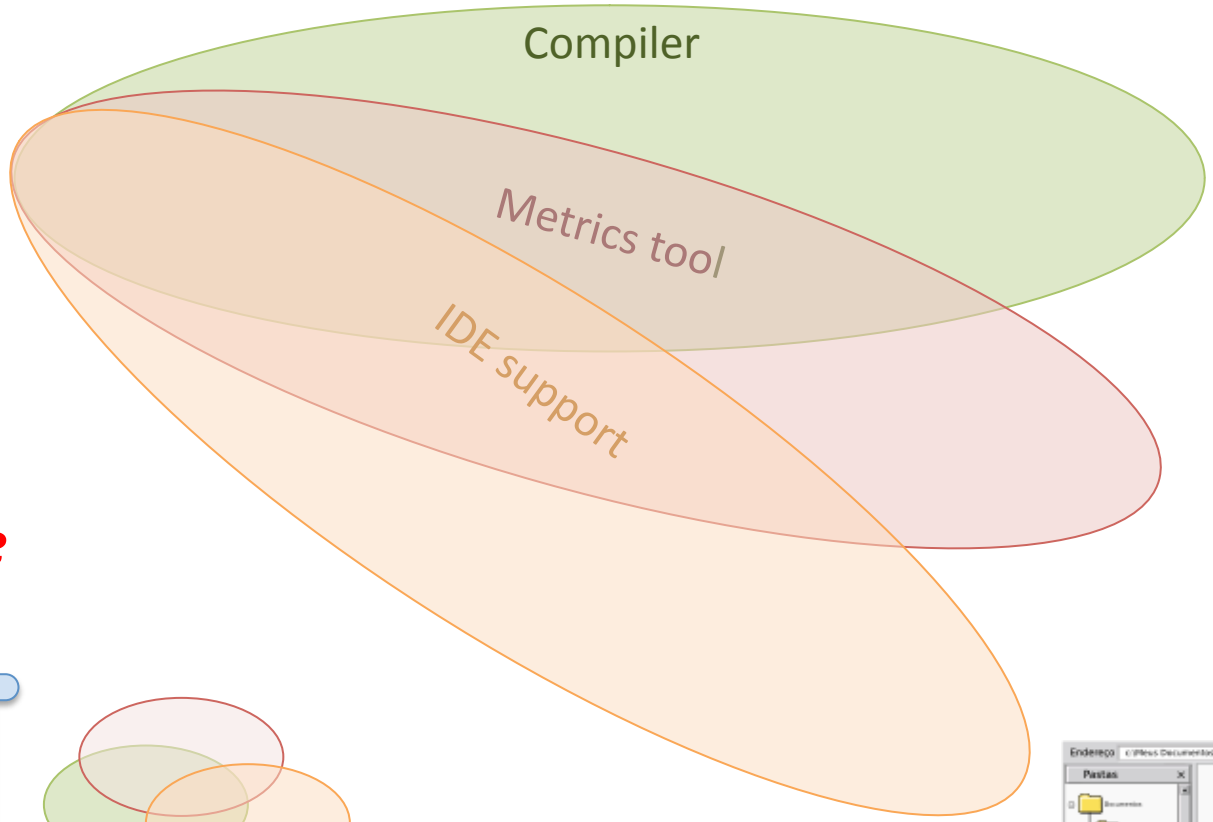
- Object-orientation (Java as host language)
- Aspect-oriented programming / Open classes
- Attribute grammars [Knuth 1968]
- Higher-order attributes [Vogt et al. 1989]
- Reference attributes [Hedin 2000]
- Context-dependent transformations [Ekman and Hedin 2004]
- …

Applications

- JModelica.org, Optimica
- JastAddJ (extensible Java compiler)
- …

Why extensible compilers?

```
a = new A
x = a.m()
```



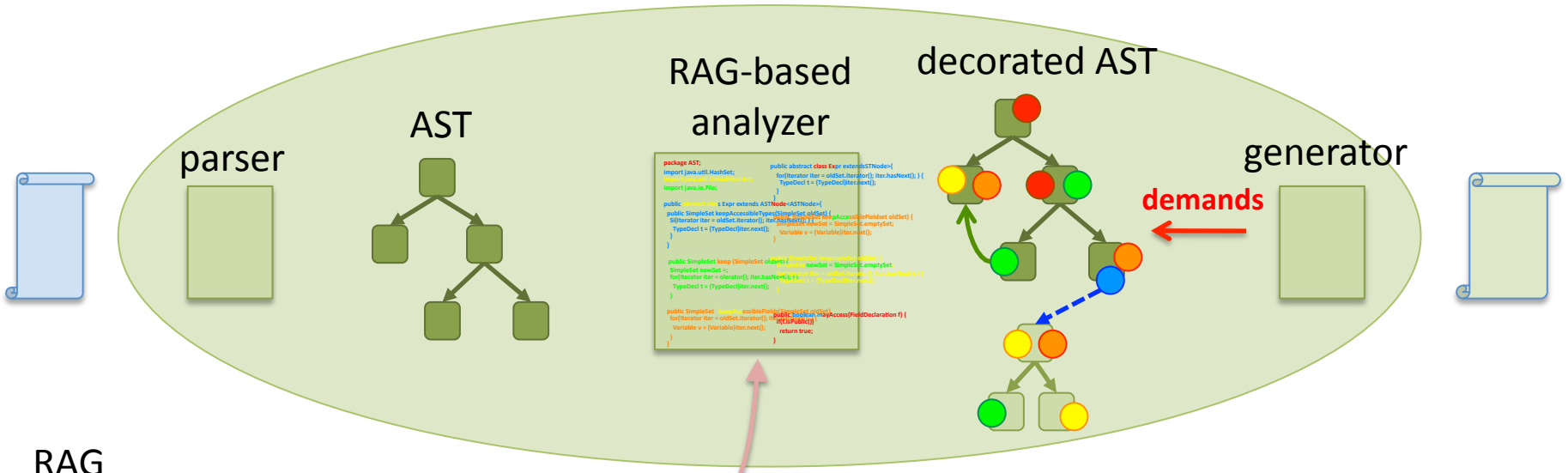

```
PUSH a_cstr
CALL
PUSH 7
STOREL
POP
PUSH a_vtbl
PUSH 4
CALLI
...
```

Extend the language

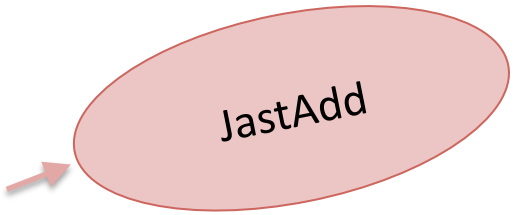
```
a = new A[T]
x = a.m()
```


42

Modularizing the compiler using JastAdd



RAG



Decorations (attributes)

- defined by equations, possibly circular
- can be references—AST becomes a graph
- can be higher-order: new ASTs
- evaluated on demand—cached for efficiency
- arbitrary modularization of individual definitions

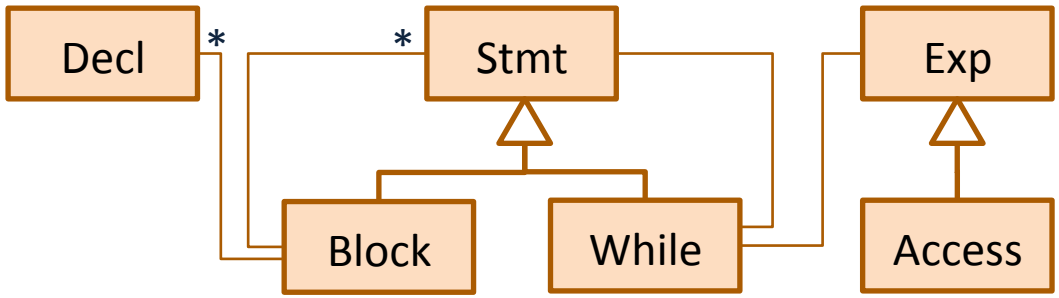
JastAdd programming mechanisms

OO: Classes, inheritance, overriding

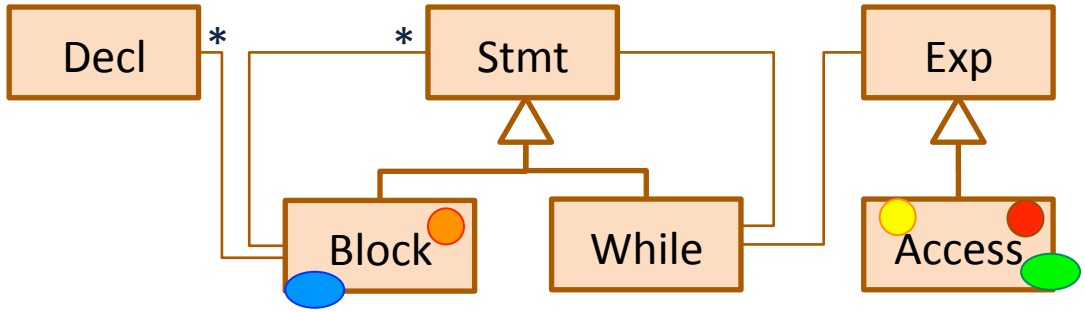
```
abstract Stmt;
abstract Exp;
abstract Decl;
While : Stmt ::= Exp Stmt;
Block : Stmt ::= Decl* Stmt*;
Access: Exp ::= ID;
```

Open classes: Inter-type declarations

```
T1 Exp.f;
T2 Exp.m() {  … }
T2 Access.m() { … }

Decl implements I;
```

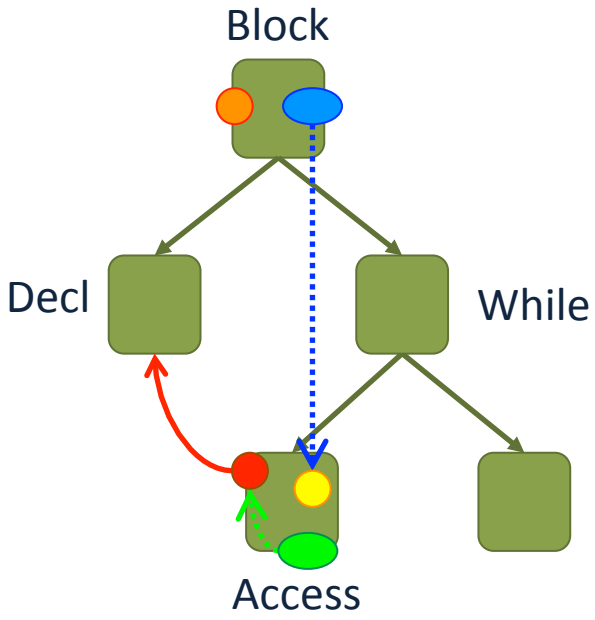


JastAdd programming mechanisms

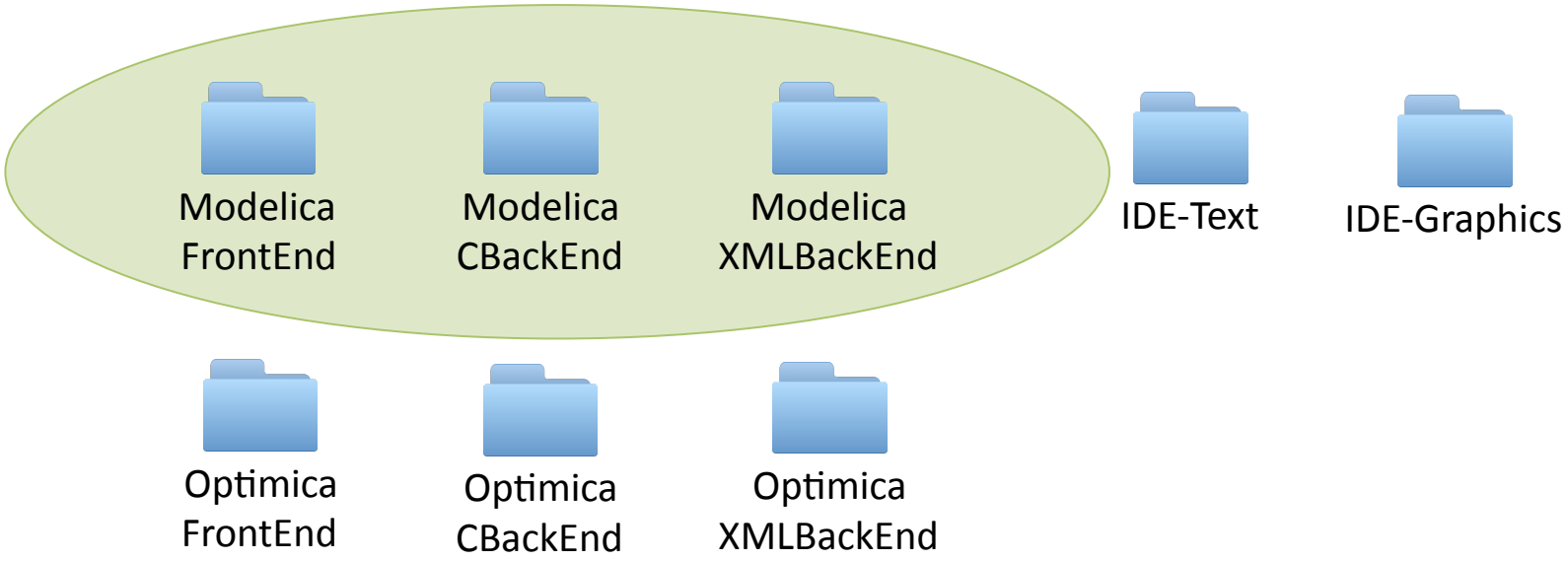


AGs: Synthesized and Inherited attributes

```
syn Decl Access.decl;
inh Decl Access.lookup(String s);
inh Decl Block.lookup(String s);
eq Access.decl = lookup(ID);
eq Block.stmts.lookup(String s) {
  res = decls.locals(s);
  if (res != null) return res;
  return lookup(s);
}
```



JModelica.org components



Compiling Modelica

```
model Bike
  Wheel frontwheel:
  Wheel backwheel;
end Bike;

model Wheel
  replaceable Brake brake;
end Wheel;

model Brake
end Brake;

model DiscBrake extends Brake
  Real discTemp;
end DiscBrake;

model DrumBrake extends Brake
end DrumBrake;

model MyBike extends Bike
  (frontwheel(redeclare DiscBrake brake),
  (backwheel(redeclare DrumBrake brake));
equation
  assert(frontwheel.brake.discTemp > 300,
    ”Alarm: front wheel temperature too high”);
end MyBike;
```

Key compilation analyses:

- Name analysis
- Type analysis
- Building the instance hierarchy

Challenge:

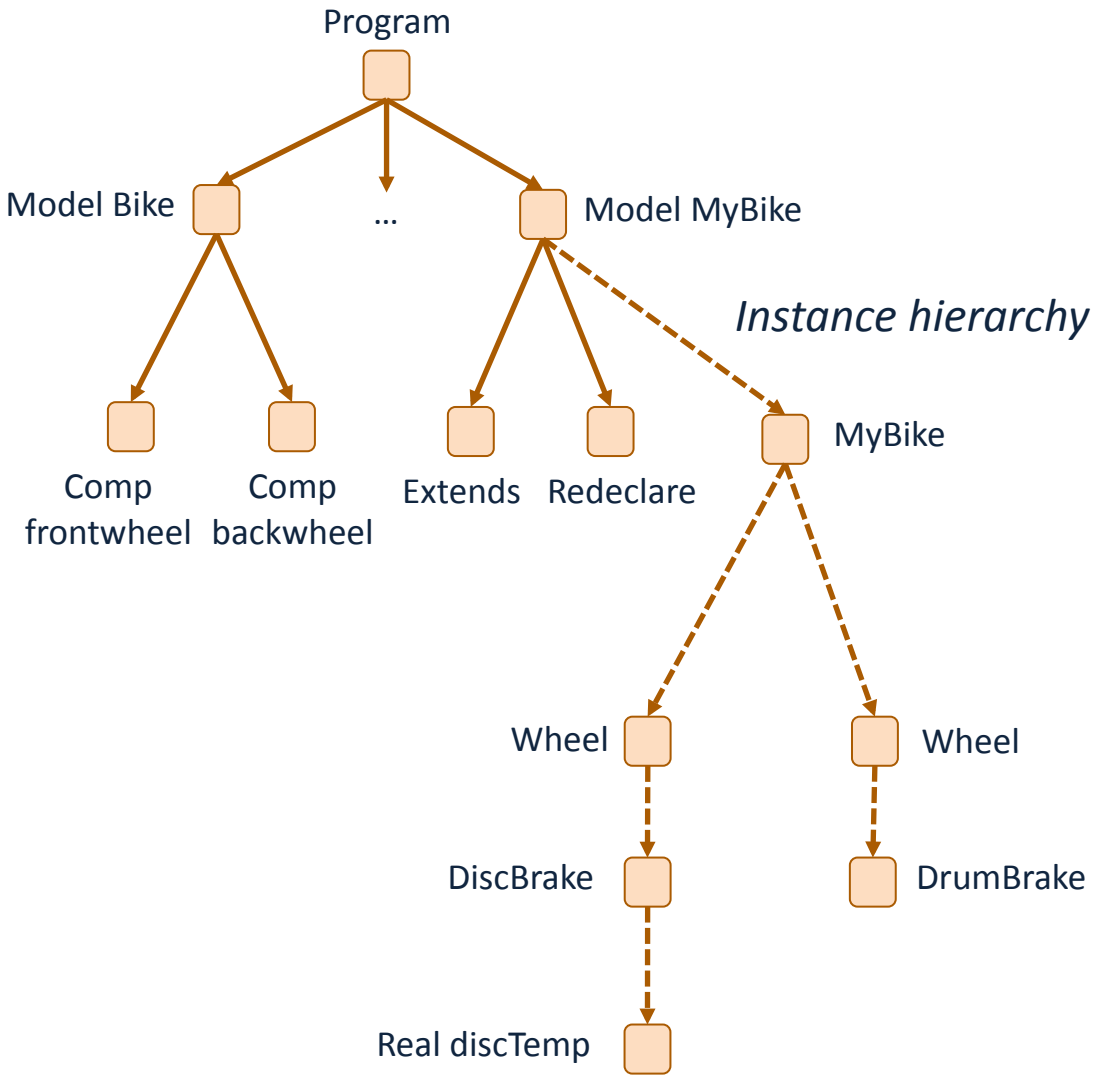
- Analyses are *interdependent*

JastAdd solution:

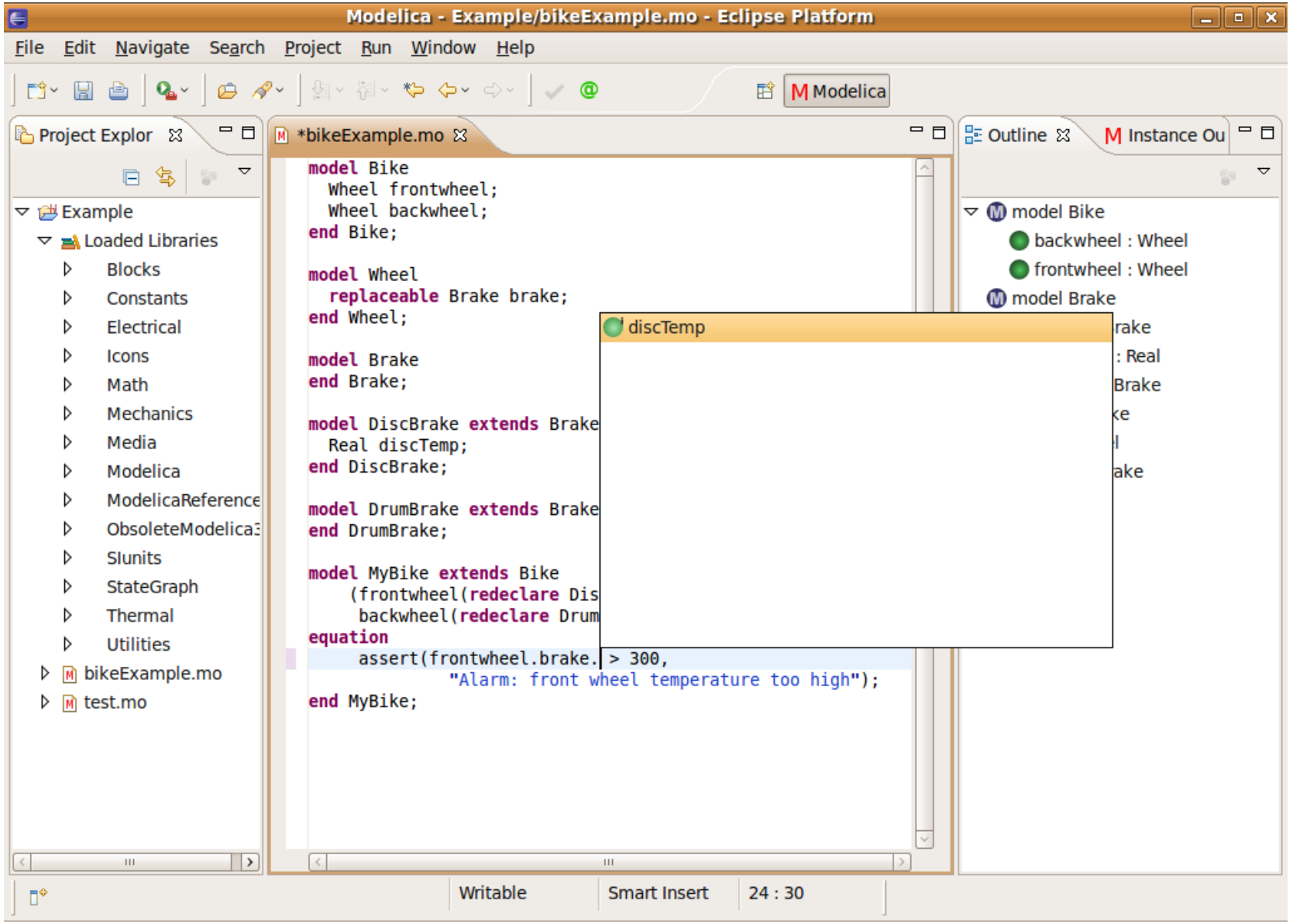
- instance tree — higher-order attributes
- automatic interleaving of the analyses

Compiling Modelica

```
model Bike
  Wheel frontwheel:
  Wheel backwheel;
end Bike;

model Wheel
  replaceable Brake brake;
end Wheel;

model Brake
end Brake;

model DiscBrake extends Brake
  Real discTemp;
end DiscBrake;

model DrumBrake extends Brake
end DrumBrake;

model MyBike extends Bike
  (frontwheel(redeclare DiscBrake brake),
  (backwheel(redeclare DrumBrake brake));
equation
  assert(frontwheel.brake.discTemp > 300,
    ”Alarm: front wheel temperature too high”);
end MyBike;
```

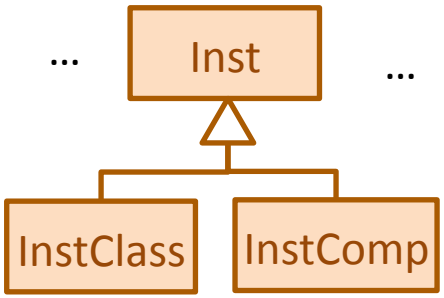


IDE name completion



Extending the compiler with name completion

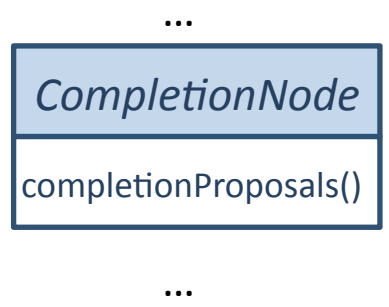
Compiler



Modelica Name Completion

```
Inst implements CompletionNode;

public ArrayList Inst.completionProposals() {
  return completionNodes();
}

syn ArrayList Inst.completionNodes();

eq InstClass.completionNodes() =
  … access compiler attributes …

eq InstComp.completionNodes() =
  … access compiler attributes …
```

IDE framework



Optimica: an extended language

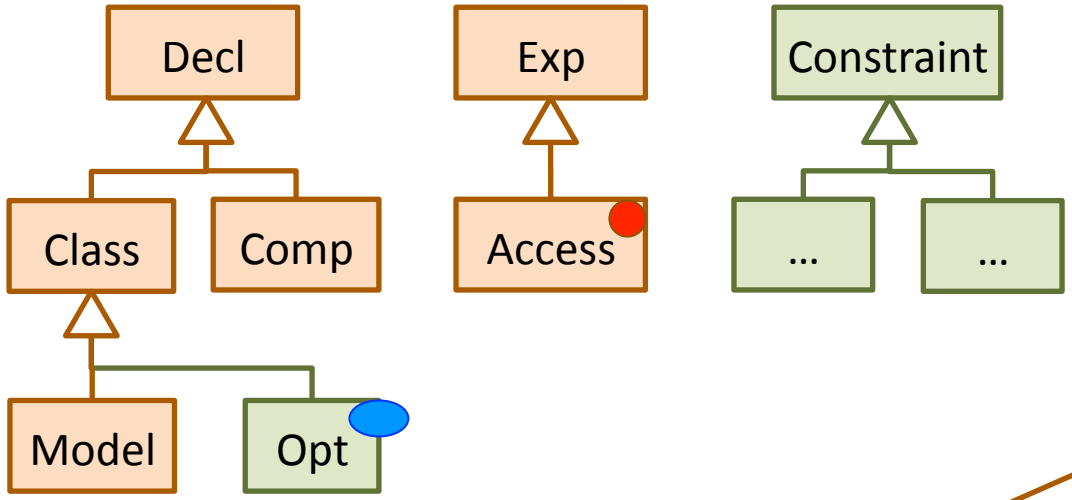
```
model Car
  Real x(start=0);
  Real v(start=0);
  input Real u;
equation
  der(x)=v;
  der(v)=u;
end Car;
```

Modelica code

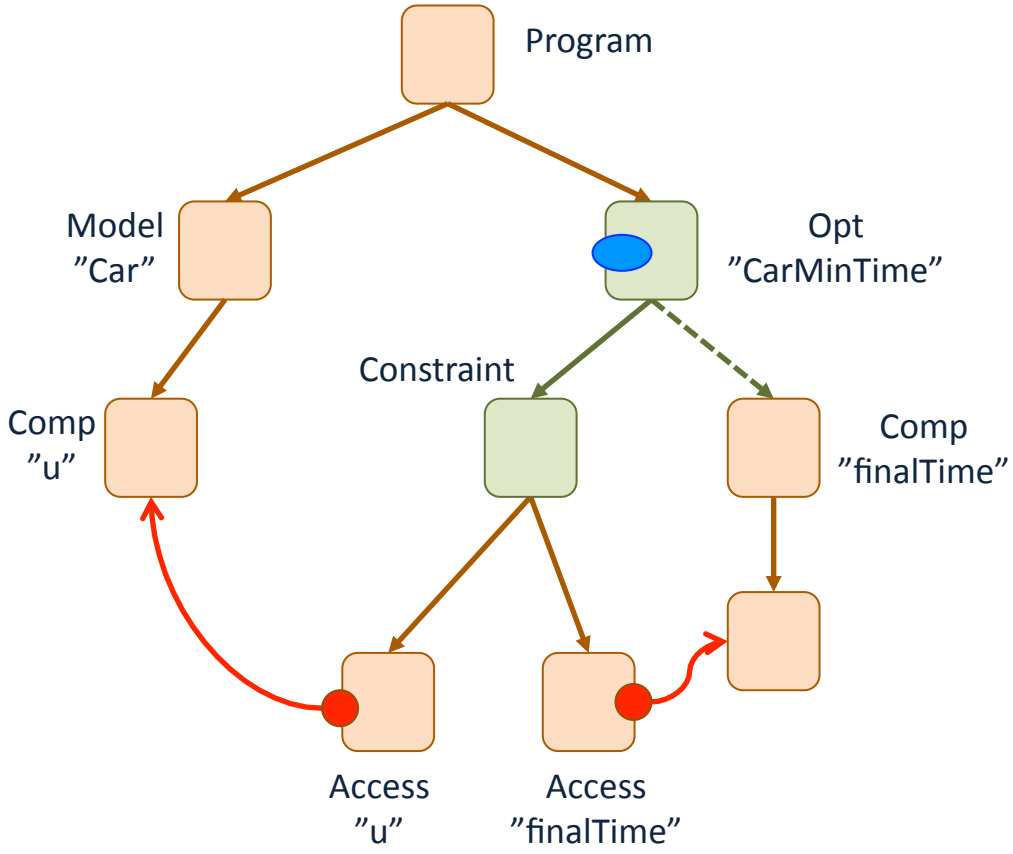
```
optimization CarMinTime (
            objective=finalTime ,
            startTime=0,
            finalTime(free=true, initialGuess=1))
      Car car(u(free=true, initialGuess=0.0));
constraint
      car.x(finalTime)=1;
      car.v(finalTime)=0;
      car.v<=0.5;
      car.u>=-1;
      car.u<=1;
end CarMinTime;
```

Optimica code
- extends Modelica with new syntax
- and changed semantics

Extending Modelica to Optimica




```
Opt: Class ::=
        ClassModification*
        Constraint*
        /predefs:Comp*/;

eq Opt.predefs =
        new List<Comp>()…;

eq Opt.child.lookup(String s) =
        res = predefs.locals(s);
        if (res!=null) return res;
        return super(s);
```



Ongoing and future work

- Incremental updating
- General IDE support
- Graphical editing
- Performance
- Higher-level specification

Conclusions

JModelica.org, a great case for JastAdd!

For more information, see jastadd.org

Thank you!

Questions?