ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

SANDS Tools on the WWW

Christophe Buffard

Swiss Federal Institute of Technology in Lausanne
Software Engineering Laboratory

June 1996

Plan

The SANDS distribution

The specification compiler

Available Tool

Available tools on Sun OS 4 are:

coopn2pl
coopncheck
coopnformat
coopmpp
coopnsimul

Available tools on Sun OS 5 are:

coopn2pl
coopncheck
coopnformat
coopmpp
coopng
TTool

The distribution

The distribution is provided by the WWW lab’s server, we download the distribution that fit the right architecture.

The distribution between architecture is made of files in tar format.

Distribution Problems

Some tools are only available on Sun OS 4 or Sun OS 5

Users are supposed to configure the system by themselves

The Spec Compiler

The compiler will translate in Ada the COOPN specifications.

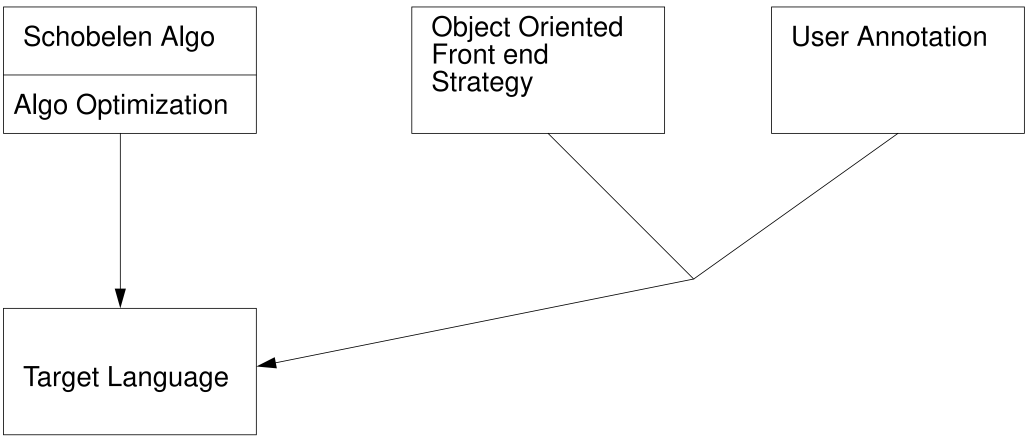
With this compiler it will be possible to execute an algebraic specification that allow immediate application for the prototyping.

This prototype can evolve by the definition of code which allow to enhance the prototype generated.

The compiler Architecture

The front end.

Algorithm used to compile specifications.



The compiler development state

- Intermediate language Definition
- Function and Classes
- Optimization
- Predefined Equality
- Grammar

Function and Class Links

Simple

- The code is generated like dynamical typing languages (Smalltalk, Self) we supposed that we can add a class method outside of the class declaration
- As default if the target language don't allow to add a class, the function will be generated as global (without redefinition)

We can force the compiler to attribute a function to a specific class.

Function and Class Links

General case

- dynamic typing languages: the operation **nat** is linked to the class *nat*.
- static typing languages: all operations are global.

```
Classes
   string(
      concat _ _ : (string) string -> string;
   );
```


Receiver Argument

The receiver argument isn't specified here (in the intermediate language) -> the compilation isn't target language dependent.

The idea is that the operations can be linked to a class without ambiguity. If the operation can't be linked to a class we made a global functions with it.

The receiver argument is by default the first argument who have the right sort. But the user can specify in an auxiliary file the receiver parameter. For the static semantic the receiver argument must be class sort of the class that we declare.

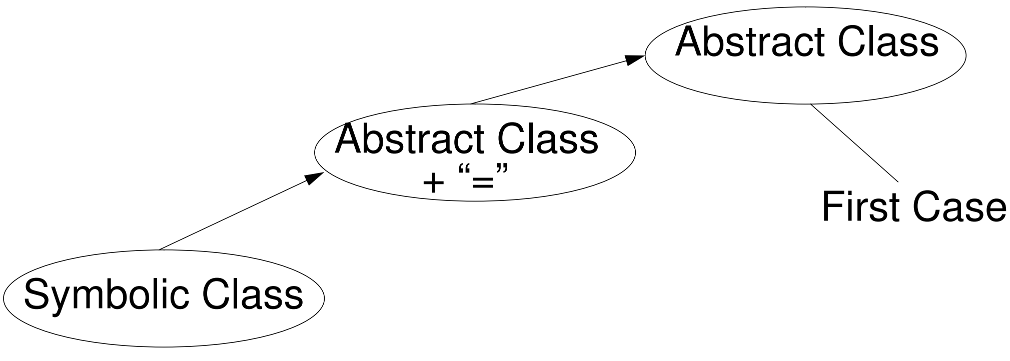
Predefined Equality

The equality is a primitive operation (method) of the abstract class.

Does the equality must be a abstract method define in each concert classes or do we need to define the equality directly in the abstract class and make it disposable for all the implementation?

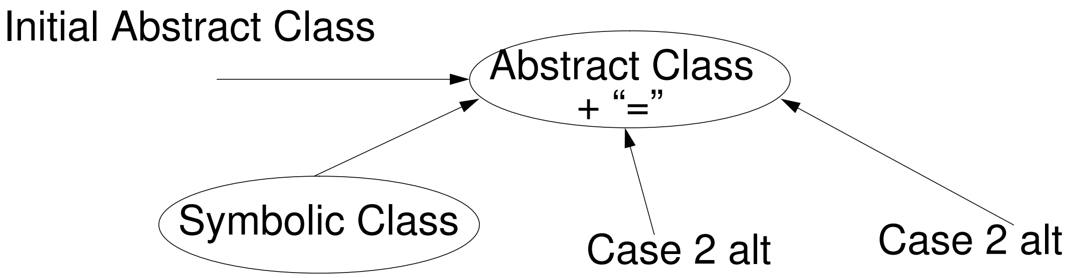
Predefined Equality

If it must be an abstract method we explicit the fact that each implementation correspond to a strict model of the specification. It's a better solution (formal point of view). The principal problem to this approach is the semantic problem in the implementation of the reverse generator.



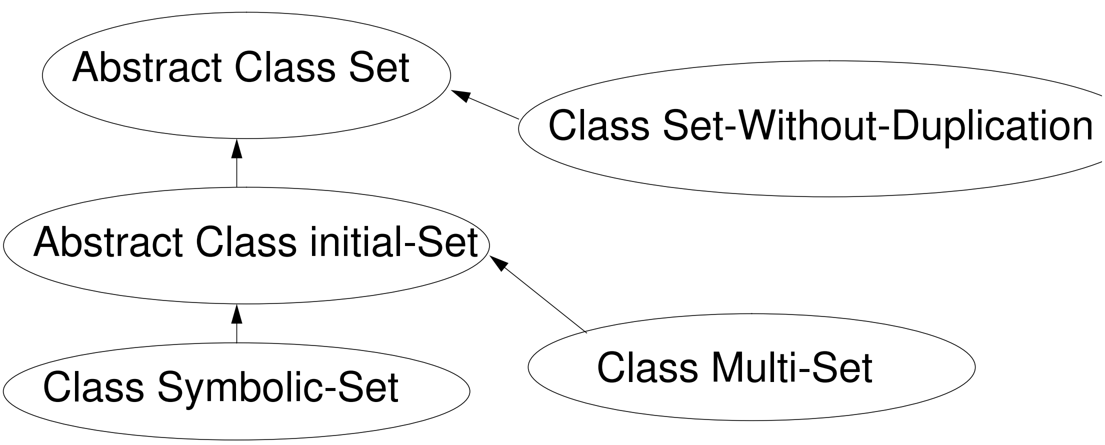
Predefined Equality

If we define directly the equality in the abstract class we adopt a point of view closer to the Object Oriented approach, and this approach will be more pragmatic.



Predefined Equality

This two chose will be disposable for the user. The first case is used when we want to simulate axioms between generators (we can simulate the set without duplication).



Future Work

- The data structure of the compiler must be validate
- The design of the complier
- The implementation in C++ and with the COOPN-lib