SANDS tools for CO-OPN

Didier Buchs, Jarle Hulaas and Pascal Racloz
ConForM group

Swiss Federal Institute of Technology in Lausanne
University of Geneva

CO-OPN

The CO-OPN specification language:

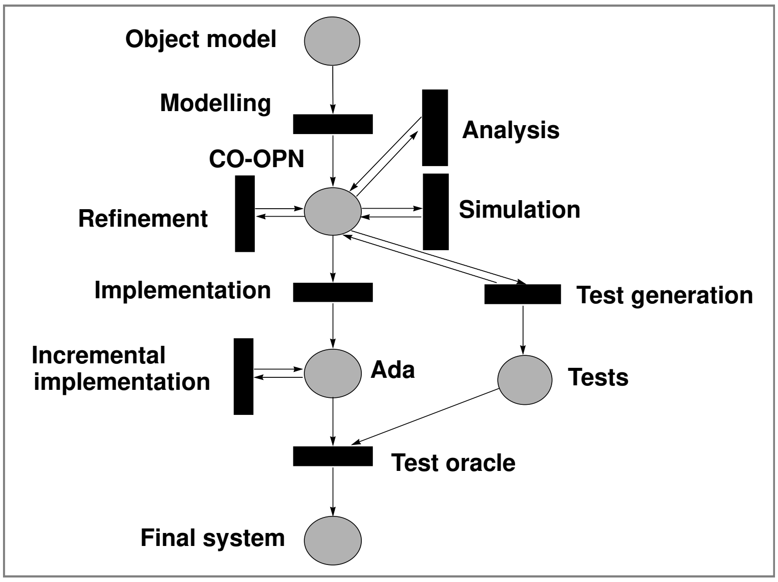
Concurrent Object - Oriented Petri Nets

Algebraic data types + Petri Nets + Modularity + Synchronizations

- **Hierarchical algebraic net modules: objects**
- **Encapsulation: methods**
- **Abstraction: synchronizations**
- **Internal behavior: internal transitions**

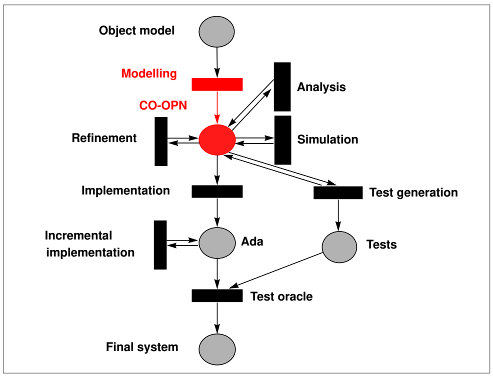
Development cycle with SANDS

SANDS: Structured Algebraic Net Development System



coopncheck, coopnformat, coopnpp

- CO-OPN language checker (syntax and static semantics)
- specification presenter for $\LaTeX$ and MIF
- specification pretty printer



coopncheck: syntax example

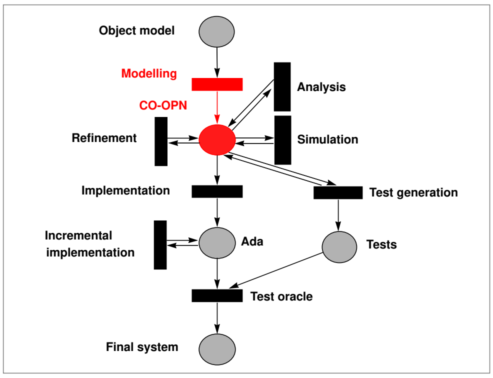
```
OBJECT DAL;
INTERFACE
 USE ADR, ListEvent, Event, Action;
 METHODS
  Act _    : action;
  DisplayModif _: listevent;
  DisplayConflict _: action;
  Transmit _ : action;
  Confirm _: action;
  Conflict _: action;
```

```
BODY
PLACES
 confirmed: listevent;
 conflicts: action;
 wait-transmit: action;
AXIOMS
 ActOk :: Consistent(a,l)=true =>
        Act a with Consult(l): -> wait-transmit a;
 ActNotOk :: Consistent(a,l)=false =>
        Act a with Consult(l: -> conflicts a;
 ...
WHERE
 a : action;
 l : listevent;
END DAL;
```

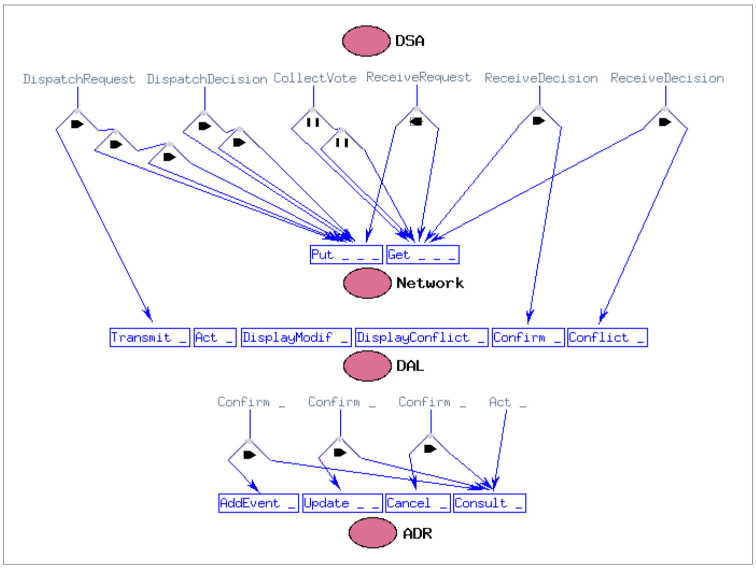

coopng

The graphical viewer:

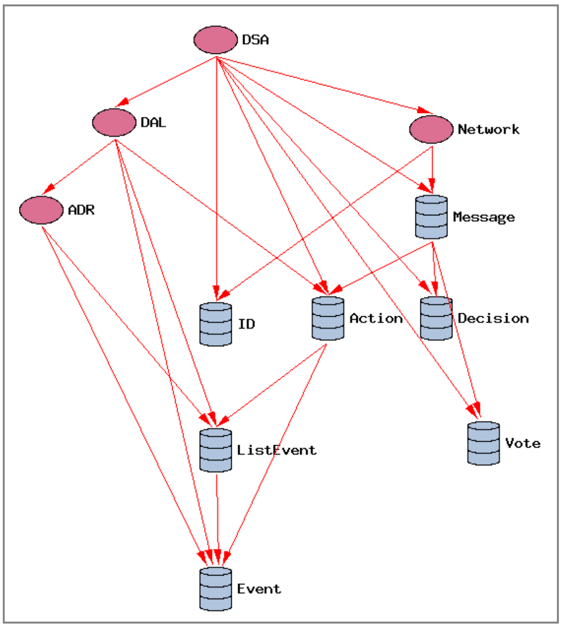
- View the global organization of the modules
  - enrichment relations
  - external methods and synchronization links
- View the internal composition of each object (algebraic nets)



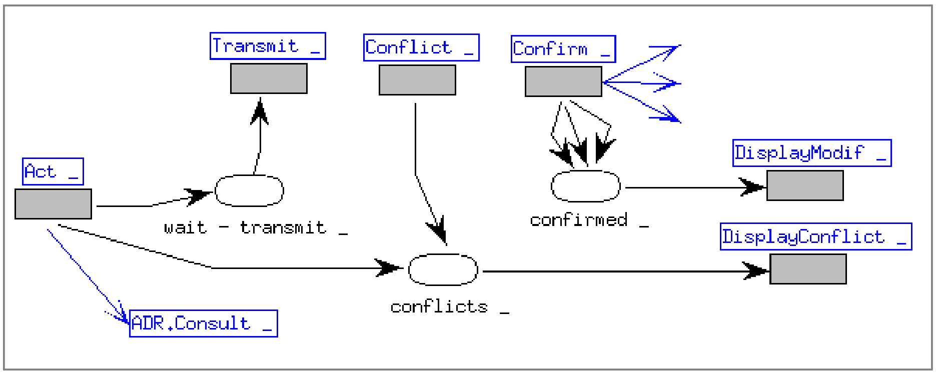
coopng: objects and synchronizations



coopng: objects, ADTs and enrichments



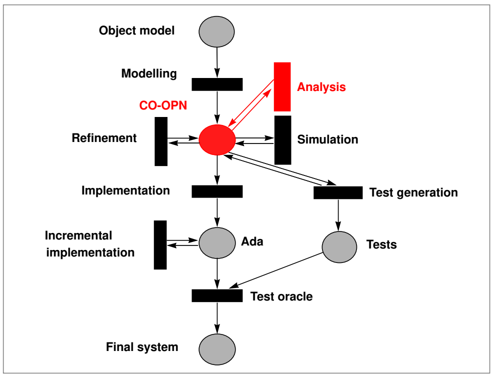
coopng: object internal structure



coopneval

The algebraic terms evaluator:

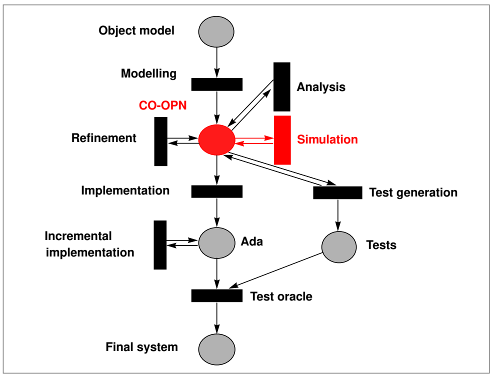
- based on Prolog
- orientation of the axioms
  ≈ rewrite rules



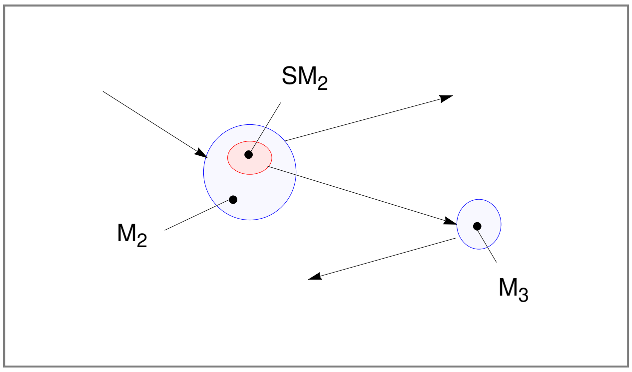
coopnsimul

The interactive object **simulator:**

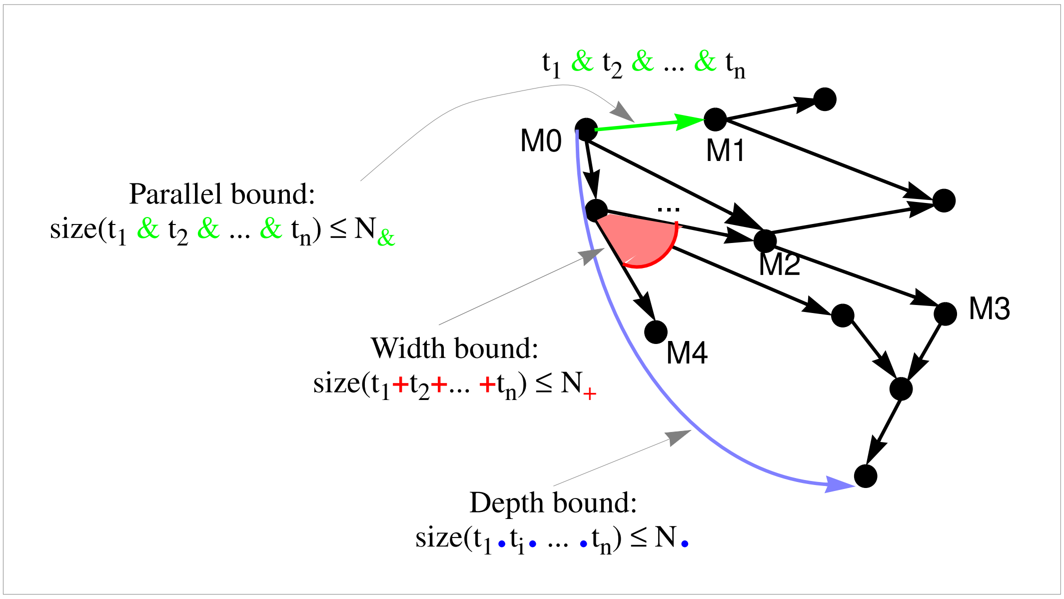
- **centralized** abstract semantics
- construction of reachability graph performed during the simulation
- possibility to set contraints on this construction w.r.t.
  - synchronization expressions
  - setting on/off methods or internal transitions



coopnsimul: generic markings



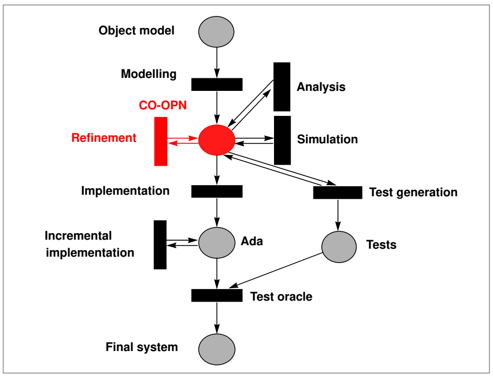
coopnsimul: kinds of constraints



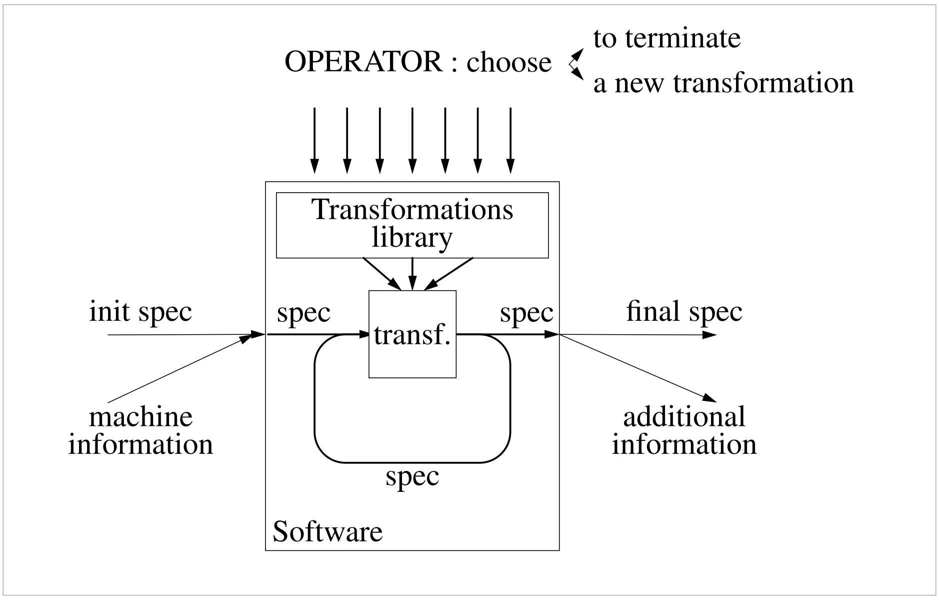
TTool

The Transformation Tool:

- transform specifications to increase parallelism
- based on rewrite techniques and relations of equivalence
- user guidance
- library of transformations:
  - branch and bound
  - divide to conquer
  - ...



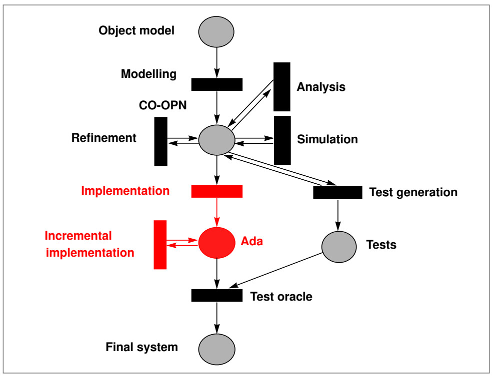
TTool: principle



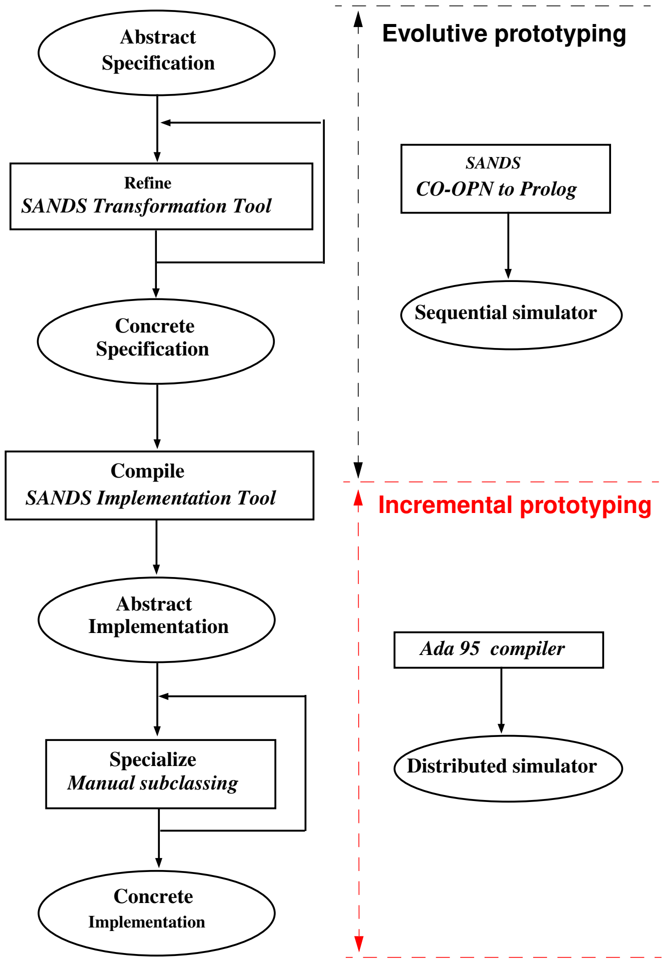
coopnc

The CO-OPN compiler:

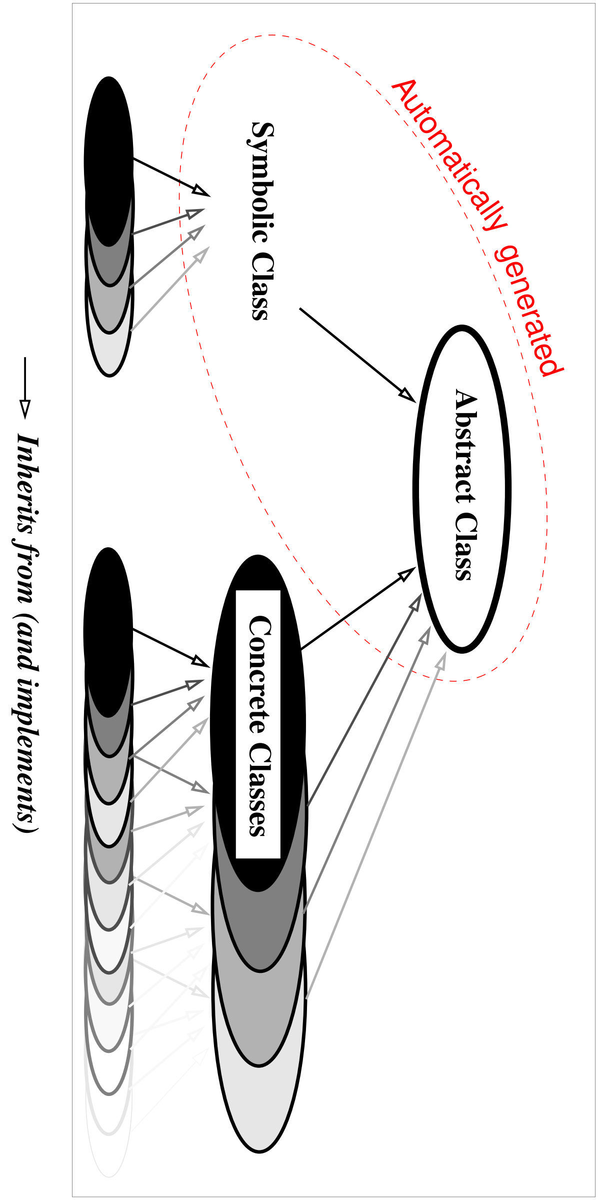
- compile specifications into distributed Ada95 prototypes
- code generated for the support of incremental (mixed) prototyping
- as distributed Prolog, but without unification
- synchronizations protected by nested transactions



The prototyping process



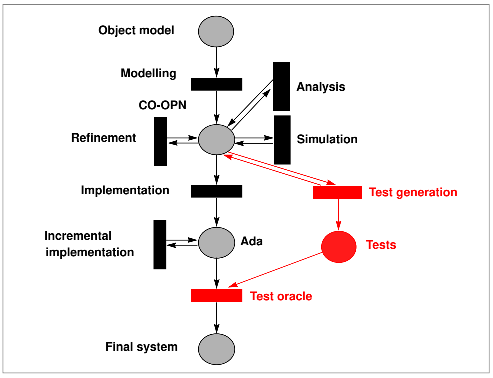
coopnc: incremental prototyping



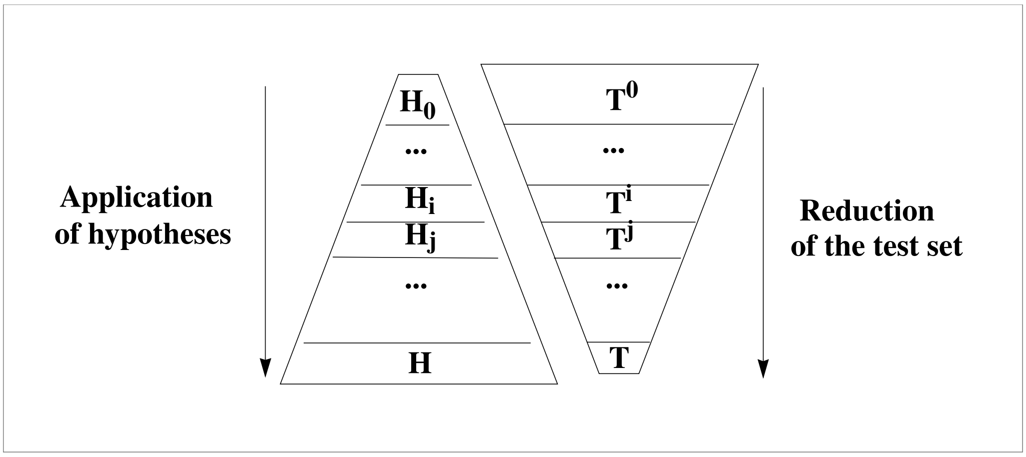
coopntest

The test set generator:

- based on observational semantics
- Hennessy Milner Logic (HML)
- reduction of the exhaustive test set:
  - regularity hypotheses on the complexity of the HML formulas
  - uniformity hypotheses on the variables



coopntest: reduction hypothesis



Future developments

- Tools for the CO-OPN/2 specification language
- Temporal logic features
- Compiler... (distributed systems)