Concepts and Realization of a Diagram Editor Generator Based on Hypergraph Transformation

Author: Mark Minas

Presenter: Song Gu

Overview

- Introduction of diagram editor
- DiaGen (diagram editor generator)
  - Overview of DiaGen
  - Common architecture
  - Free-hand editing
  - Syntax-direct editing
  - Automatic layout mechanism
- Conclusions

Introduction

What is a ***diagram editor***?

- A graphical editor which is tailored to a specific diagram
- Can “understand” the edited diagram to some extend
- Do not allow arbitrary drawing, only restrict to visual components in a certain diagram language
- Two categories: syntax-direct editing & free-hand editing

Introduction (cont.)

Syntax-direct editing:

- Each operation modifies the meaning of the diagram
- Require an internal diagram model
- These models are commonly described by a kind of graph
- Editing operations are represented by graph transformations

Introduction (cont.)

Free-hand editing:

- Allow users to directly create and manipulate any diagram
- No restrictions to predefined editing operations
- Internal diagram model is not necessary
- Diagram language can be defined by a concise graph grammar
- Need a parser to check the correctness of diagrams and analyze the syntactic structure

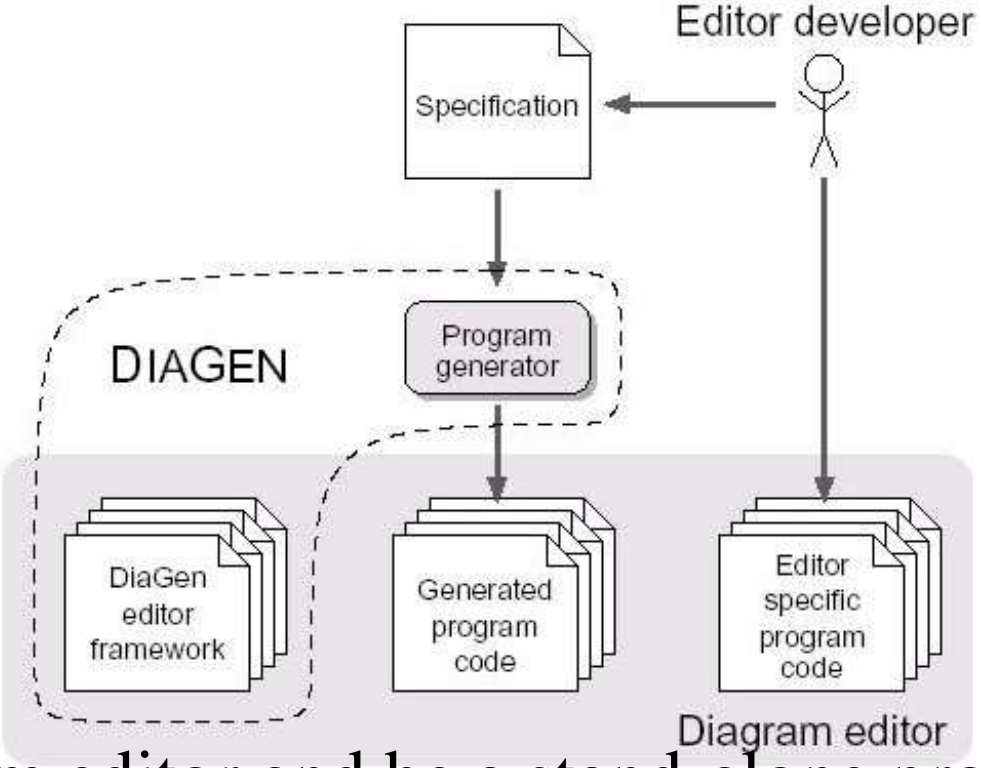
Overview of DiaGen

DiaGen is a system for easy developing of powerful diagram editors. It contains two main parts:

- A editor framework of Java classes that provide generic functionality for editing and analyzing diagrams
- A program generator that can produce Java source code for most of the functionality

Overview of DiaGen (cont.)

Developing diagram editor with DiaGen:



The Diagram editor and be a stand-alone program, or can be integrated into other software system.

Overview of DiaGen (cont.)

The key features of Diagram editor:

- Supports free-hand editing.
- Diagrams are translated into a semantic representation
- Support syntax-direct editing
- Support automatic layout

Overview of DiaGen (cont.)

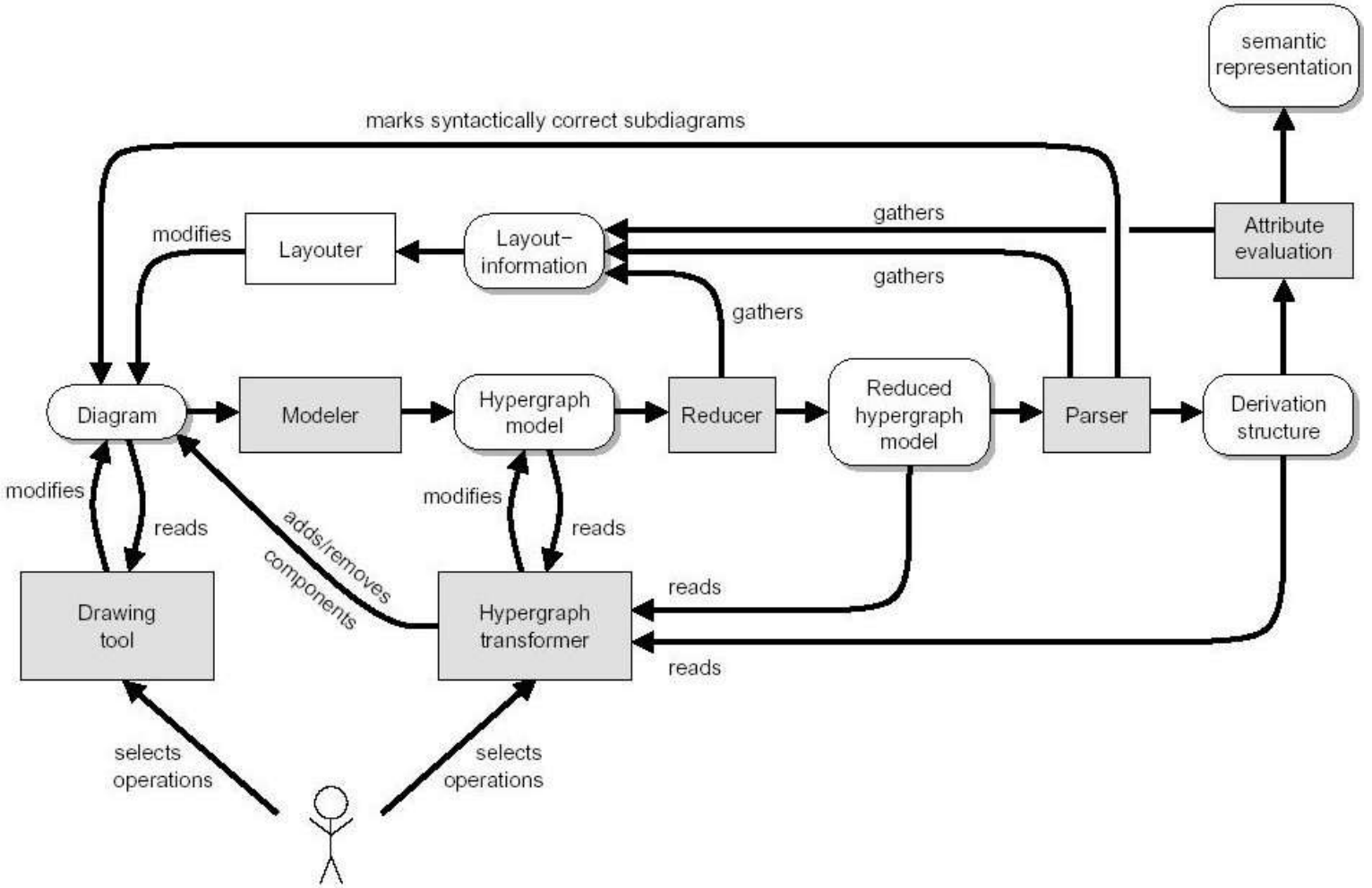


Fig. 2. Architecture of a diagram editor based on DiaGen.

DiaGen: hypergraph model

What is hypergraph:

hypergraph is used to describe a diagram as a set of diagram components, and the relationships between attachment areas of "connected" components.

- A finite set of nodes
- A finite set of hyperedges, each of which carries a type and is connected to an ordered sequence of nodes.

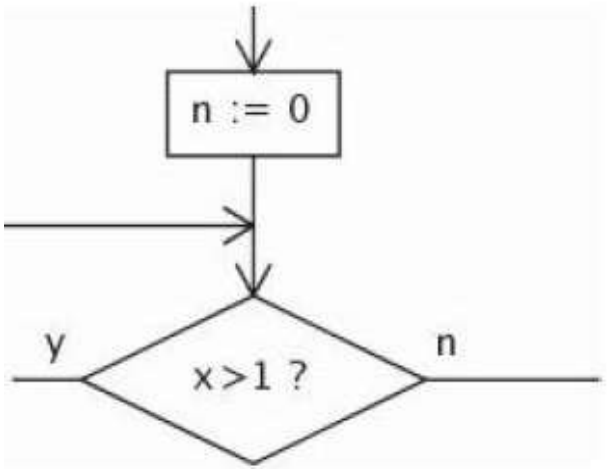
DiaGen: hypergraph model (cont.)

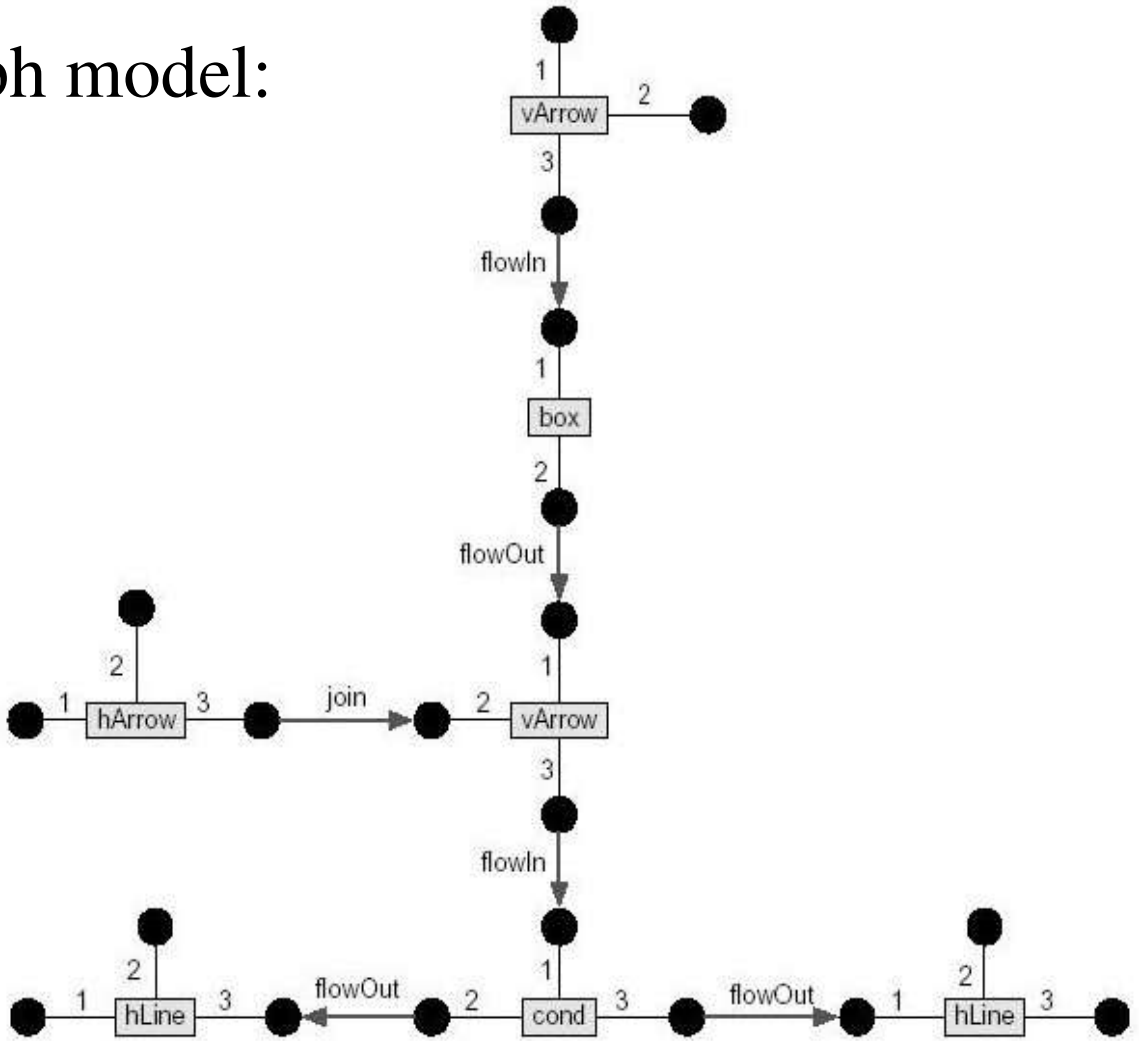
How to model diagrams with hypergraph:

- Each diagram component is modelled by a hyperedge
- Attachment areas are modelled by nodes which are visited by the hyperedge
- The sequence of visited nodes determines which attachment area is modelled by which node.

DiaGen: hypergraph model (cont.)

Example of a hypergraph model:





Overview of DiaGen (cont.)

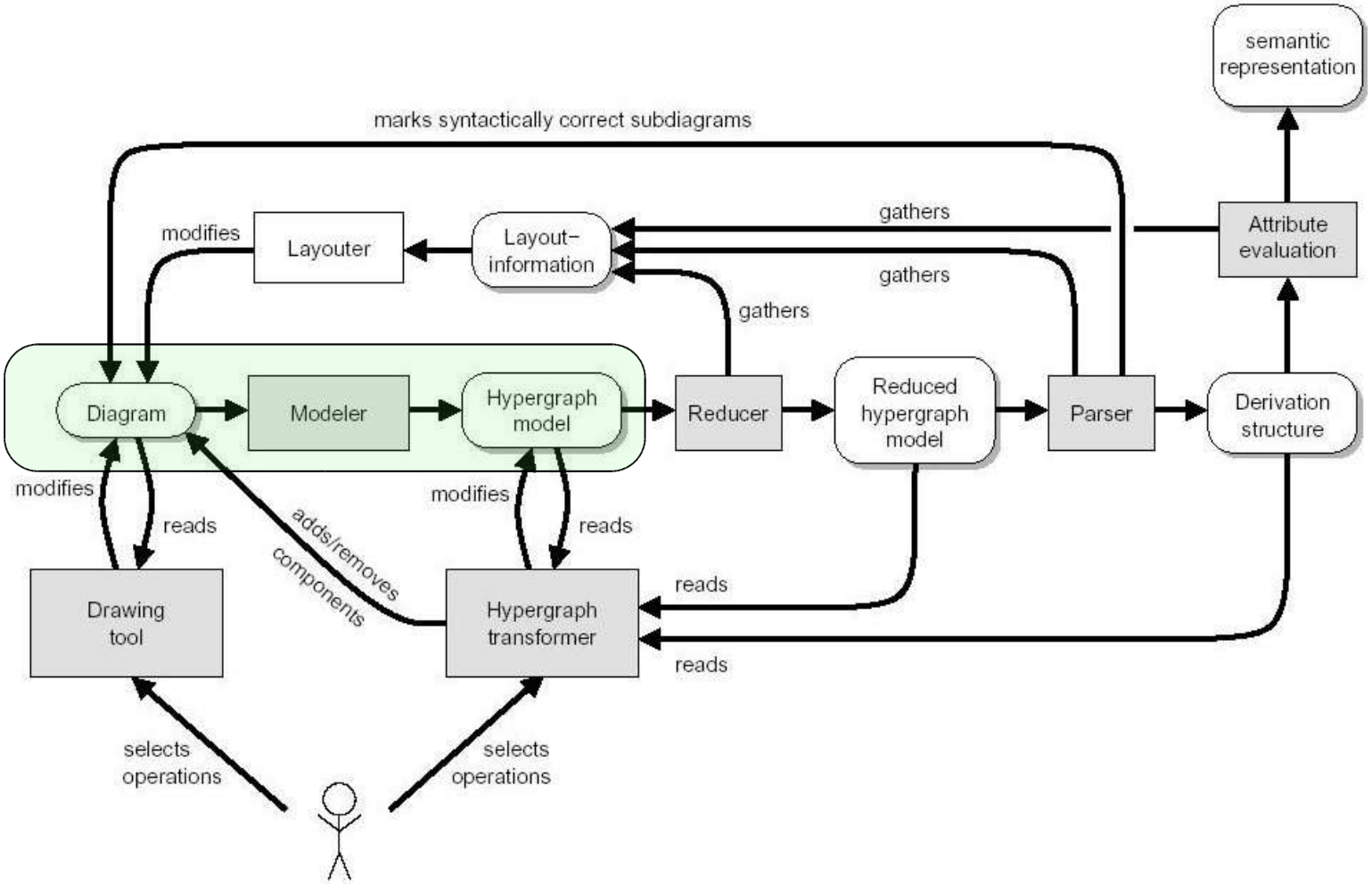


Fig. 2. Architecture of a diagram editor based on DiaGen.

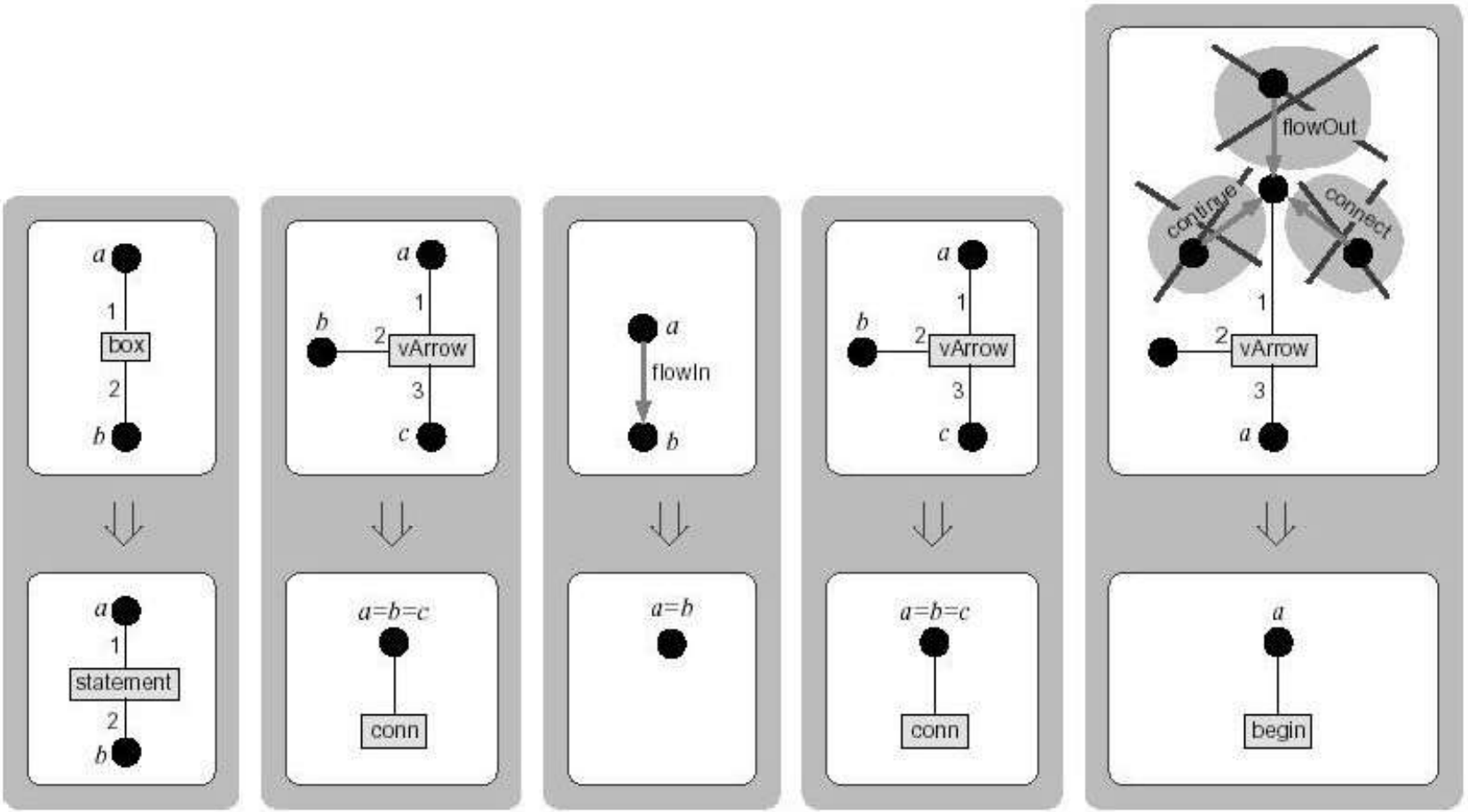
DiaGen: reduced hypergraph model

Problem: Hypergraph is too big and contains some irrelevant info. While normally the structure and meaning of a diagram are represented in terms of larger groups of components and their relationship.

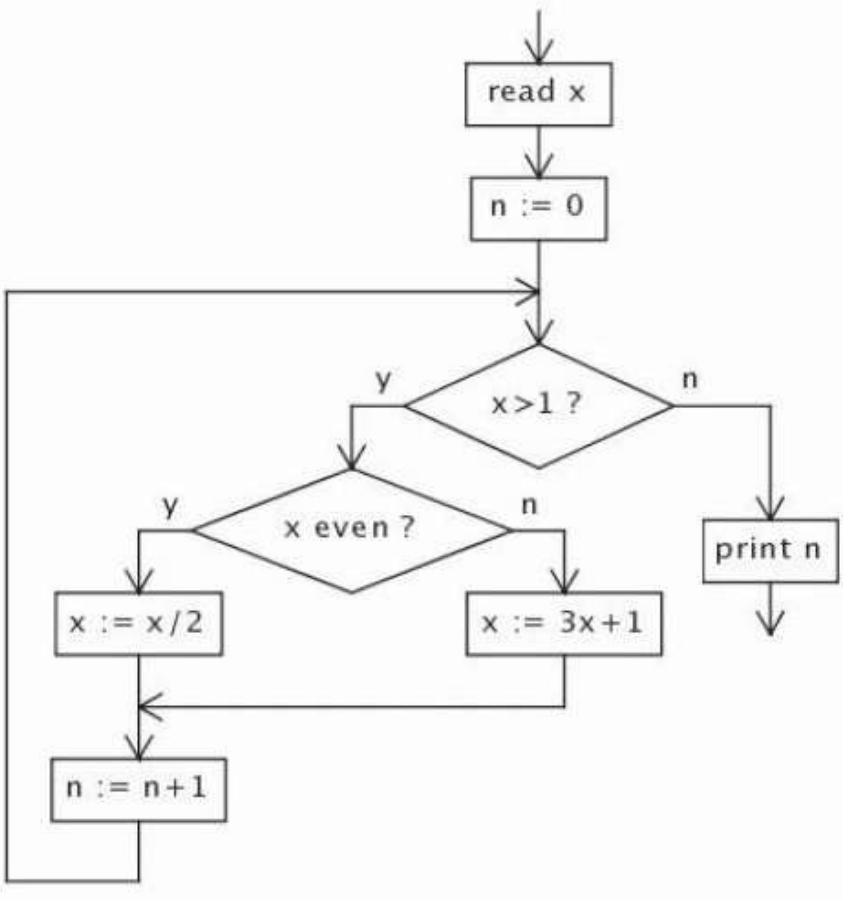
Solution: reduce the hypergraph model following the reduction rules. Each rule contains (P, R) and additional conditions. P is the pattern to search, R is the result after reduce. (kind of lexical analysis)

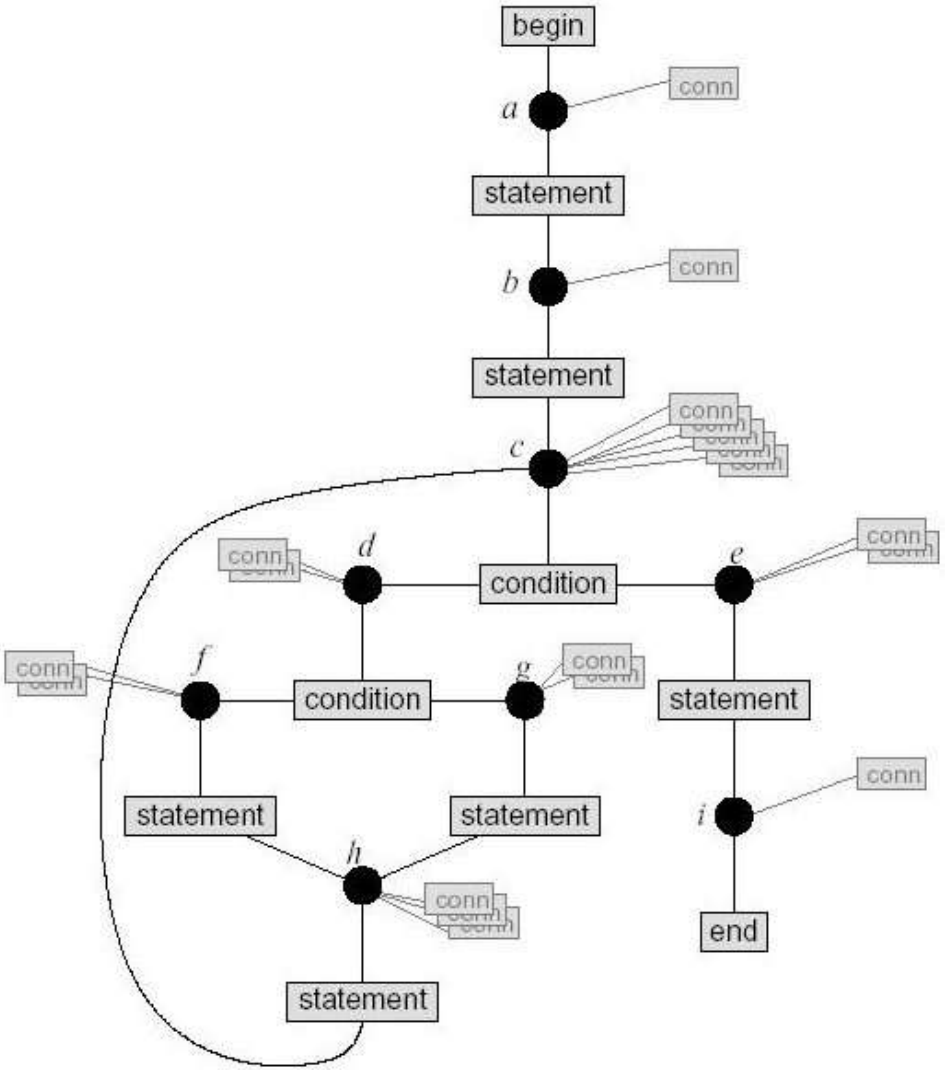
DiaGen: reduced hypergraph model

Examples: reduction rules of flowchart



DiaGen: reduced hypergraph model





Overview of DiaGen (cont.)

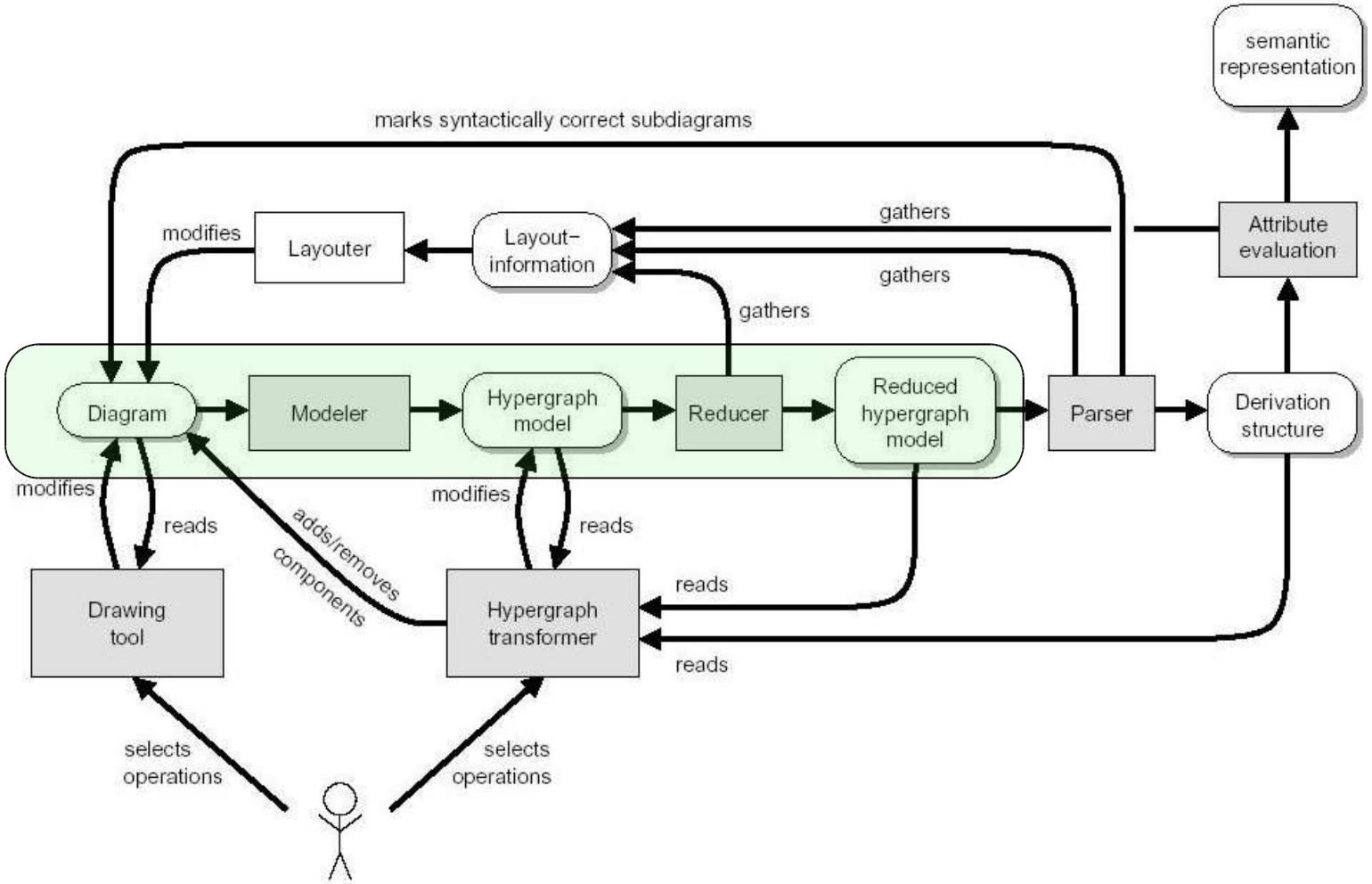
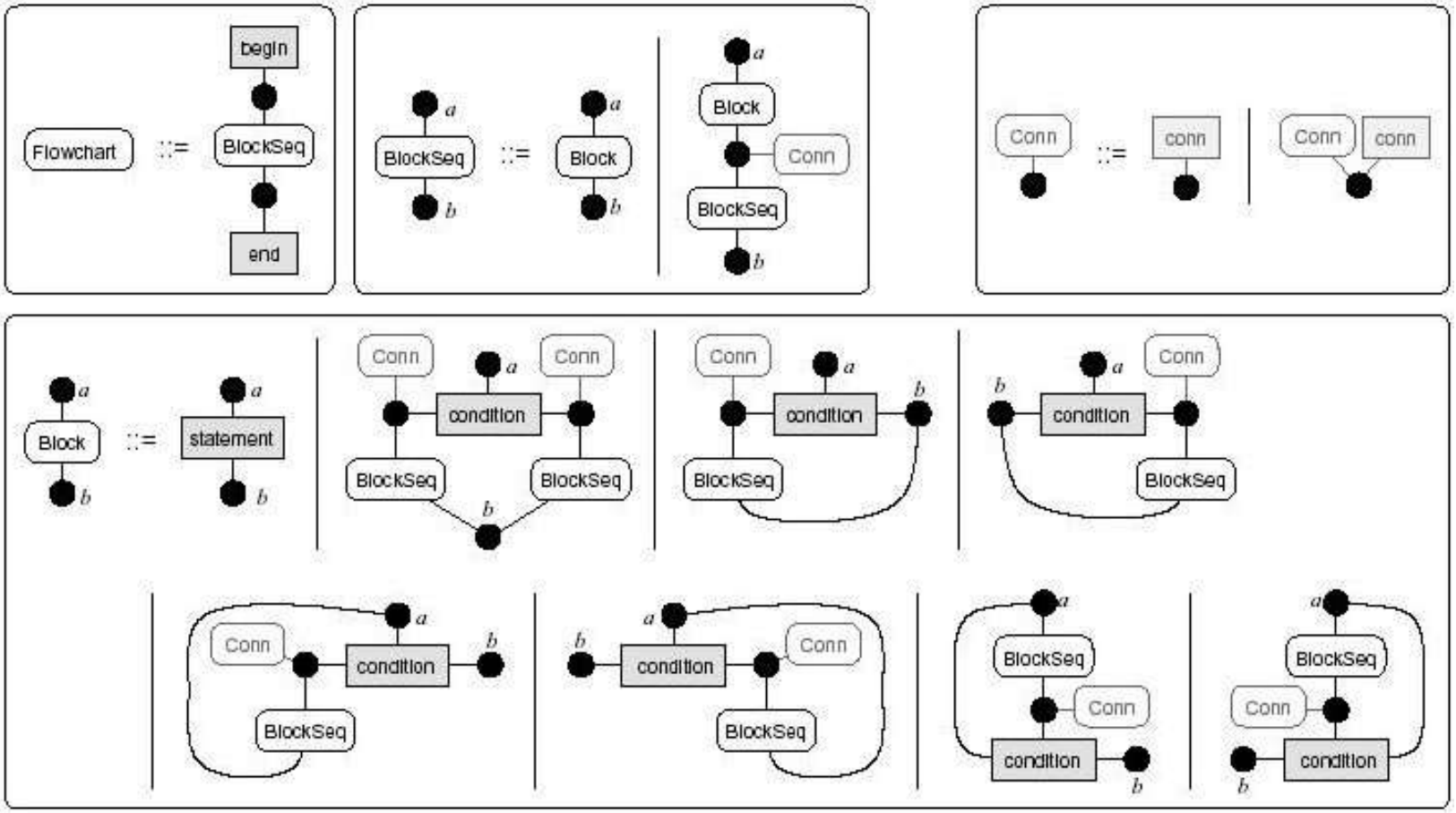


Fig. 2. Architecture of a diagram editor based on DiaGen.

DiaGen: Parser

To specify the hypergraph classes, DiaGen uses hypergraph grammars, which are similar to string grammars.



DiaGen: Parser (cont.)

Actually, DiaGen uses context-free hypergraph grammars with "embeddings".

L::= R, where R extends L by some edges and notes are embedded into the context provided by L.

- More expressive, suitable for all possible diagram languages
- Still allows for efficient parsing

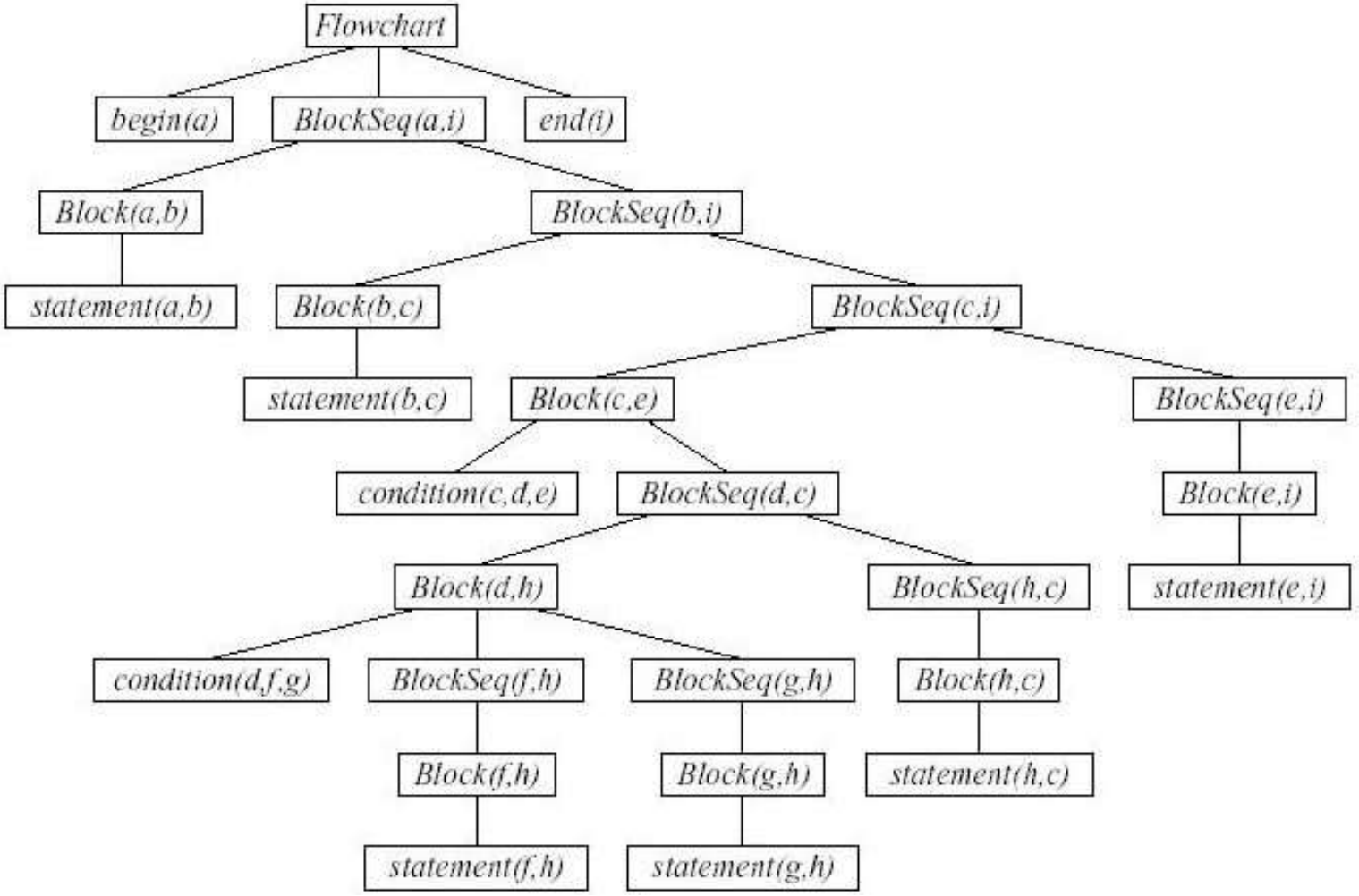
DiaGen: Parser (cont.)

The greatest feature of the parsing algorithm is the capability of dealing with diagram errors.

When errors are detected, the parser does not just reject the reduced hypergraph model, it finds out the maximum correct sub-diagrams and feedback to users by drawing them in the same color.

The results of parsing step is the derivation tree.

DiaGen: Parser (cont.)



Overview of DiaGen (cont.)

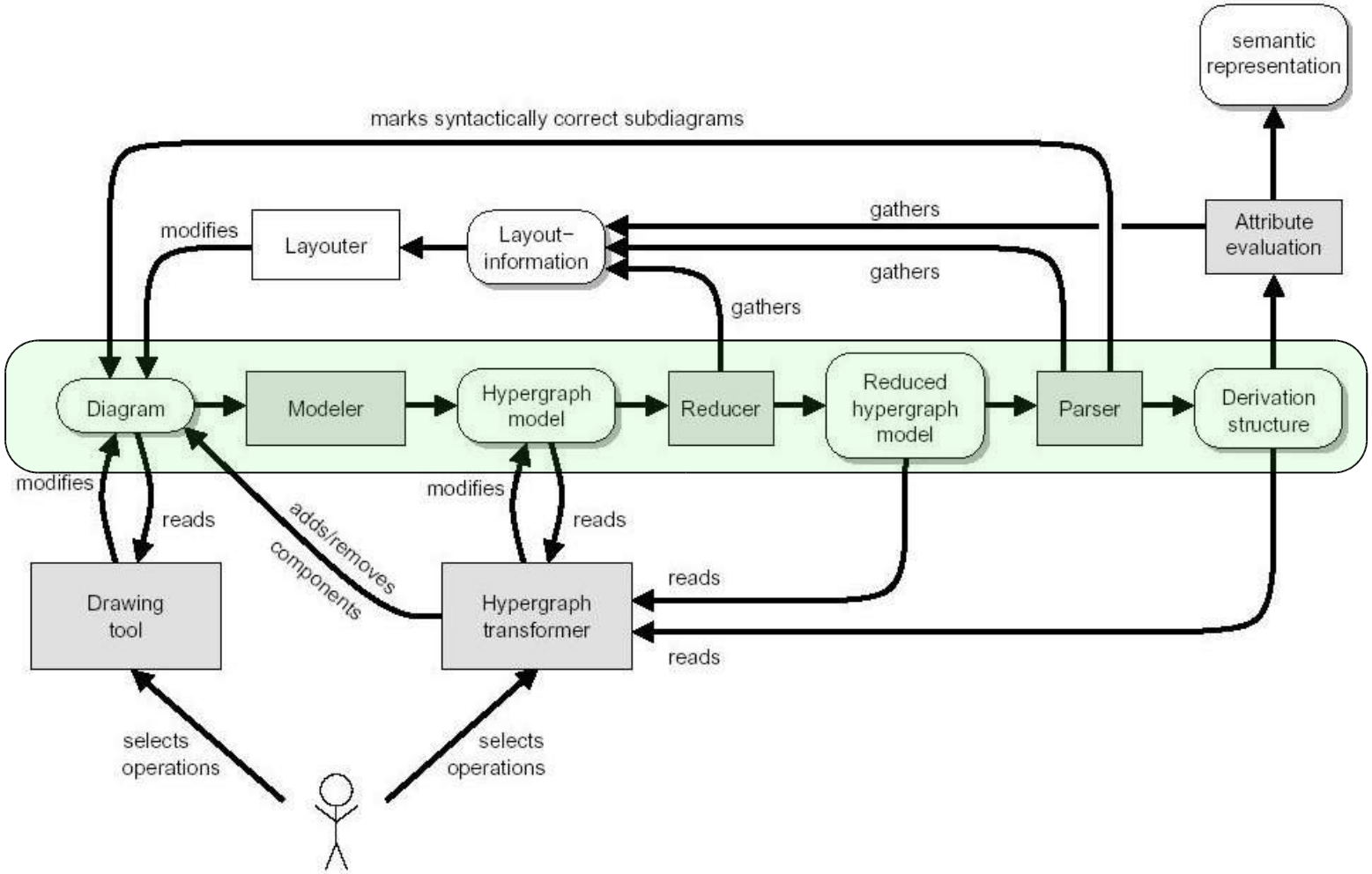


Fig. 2. Architecture of a diagram editor based on DiaGen.

DiaGen: Attribute evaluation

To translate the diagram into some data structure which is specific for the application domain, DiaGen used a common syntax-direct translation based on attribute evaluation.

- Each hyperedge that occurs in the derivation tree has a distinct number of attributes
- Grammar productions impose the rules to compute attributes values
- Evaluation mechanism will determine an evaluation order

Overview of DiaGen (cont.)

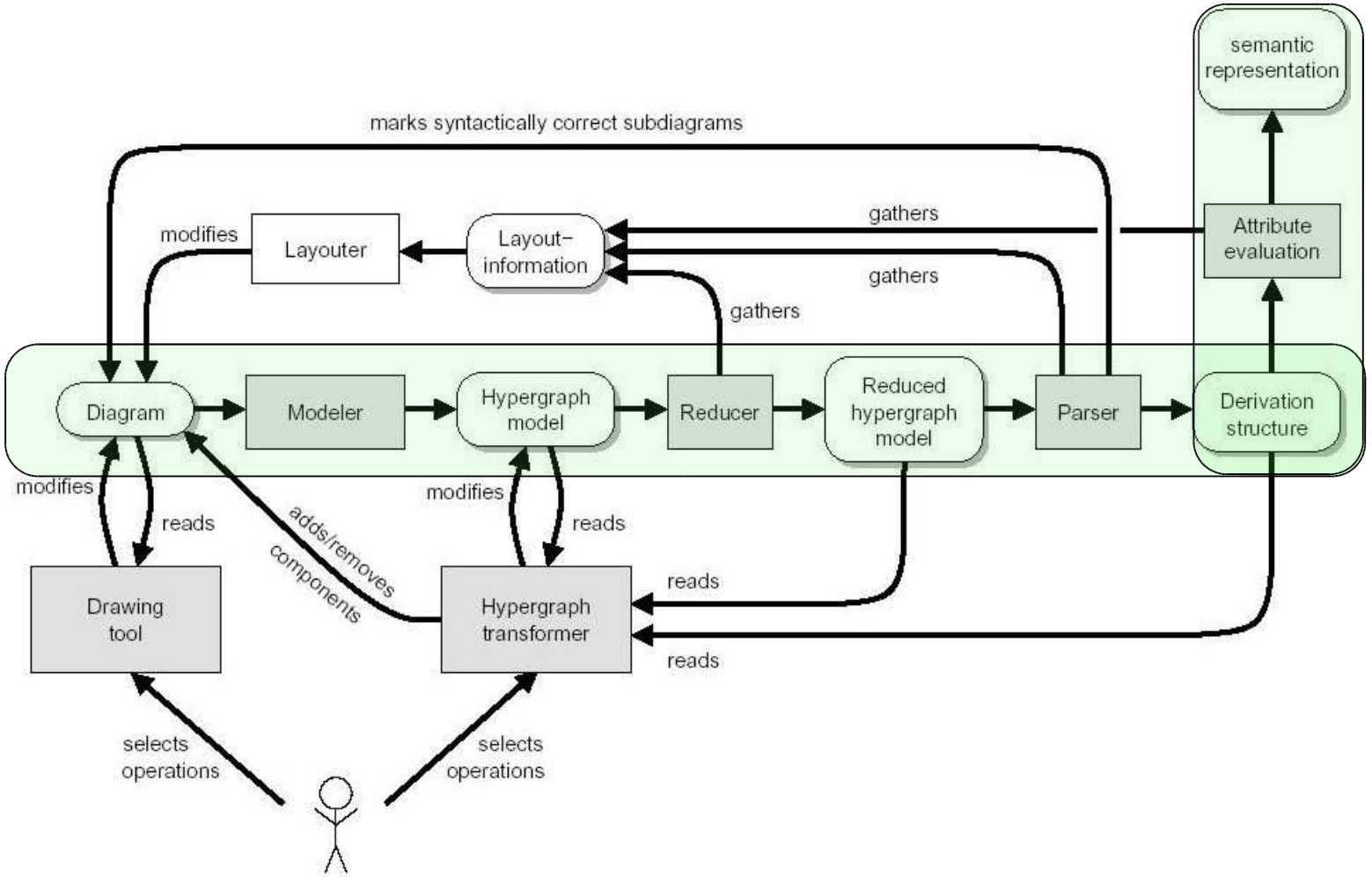


Fig. 2. Architecture of a diagram editor based on DiaGen.

DiaGen: Syntax-direct editing

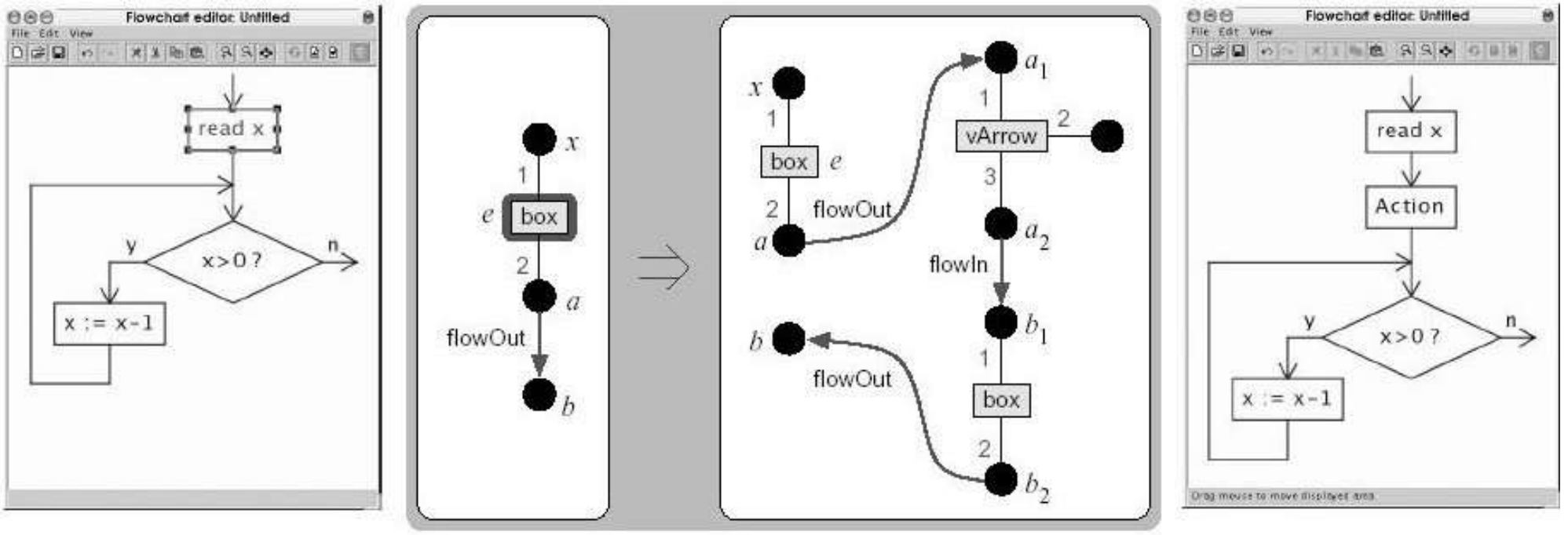
- Since DiaGen uses a internal hypergraph model, it is easy to extend to provide syntax-direct editing.

- Free-hand editing using a parser requires syntax-direct editing should not change the syntax of diagram language

- Editing operations are specified by hypergraph transformation on hypergraph model.

DiaGen: Syntax-direct editing (cont.)

Example: insert a new statement into flowchart.



DiaGen: Syntax-direct editing (cont.)

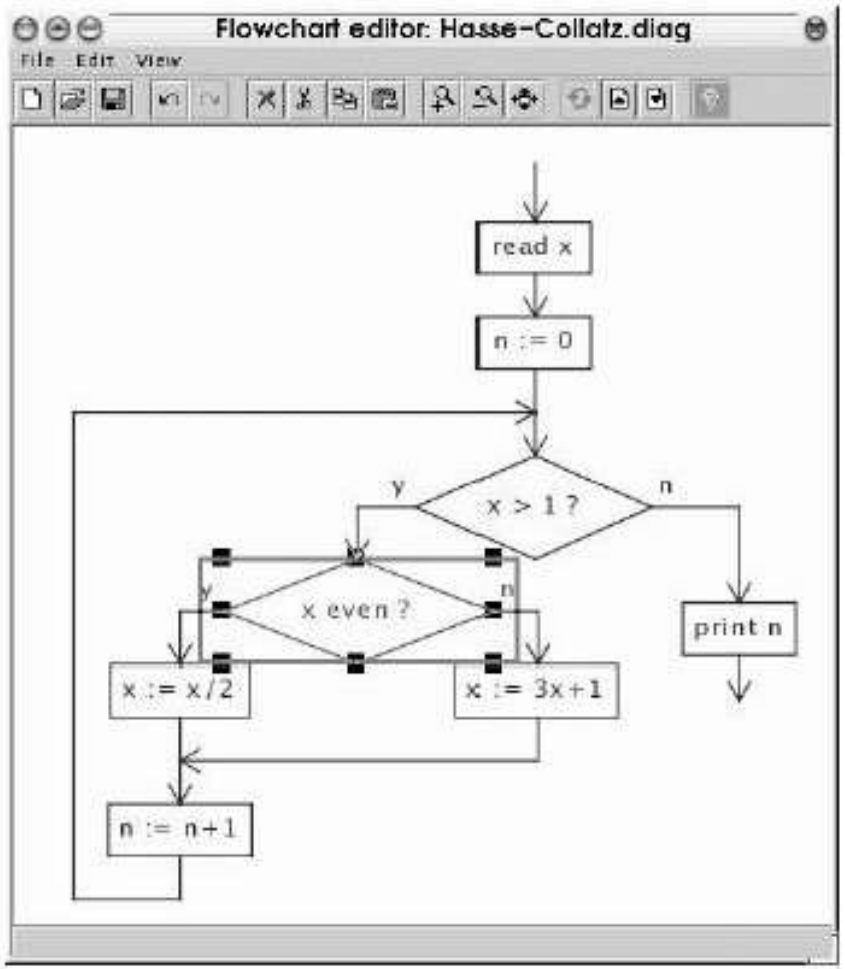
Editing operations are specified in terms of ***rules*** and ***operations.*** ie:

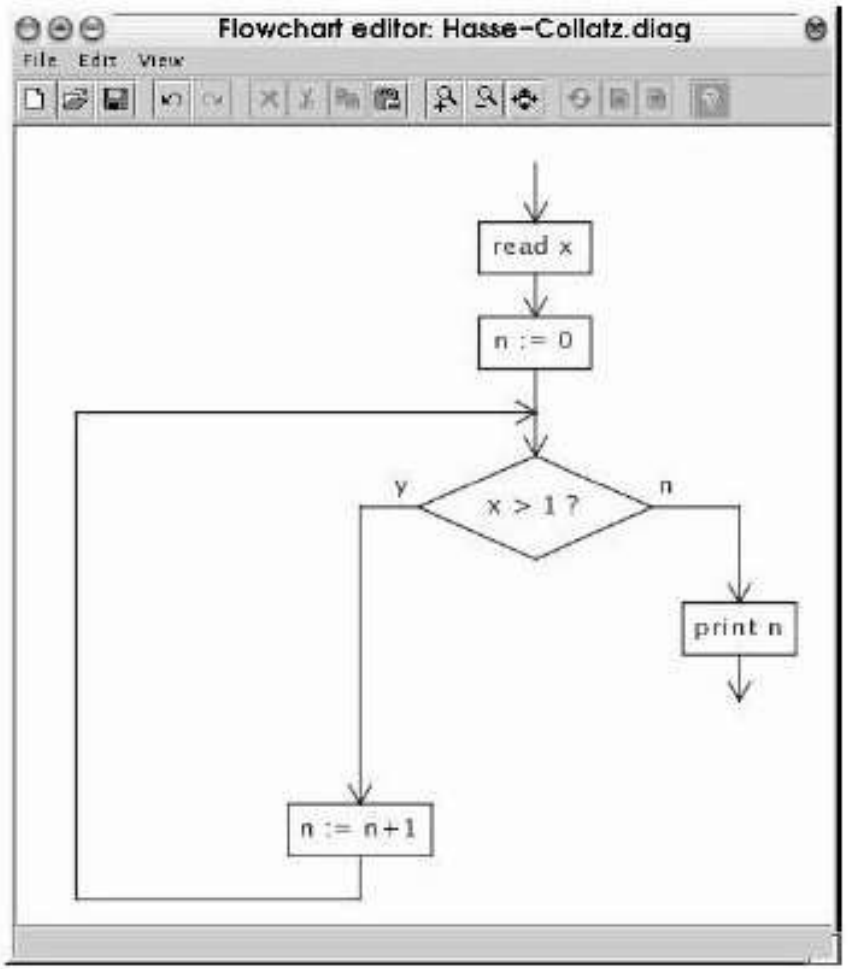
```
rule add_rule:
     box(_,a) f:flowOut(a,b) n:node(a)
     do -f
        +vArrow(a1,_,a2) { OperationSupport.createVArrow(n) }
        +box(b1,b2) { OperationSupport.createBox(n) }
        +flowOut(a,a1)
        +flowIn(a2,b1)
        +flowOut(b2,b);

operation add_stmt_after_stmt "Add statement" :
          specify box b "select statement"
          do add_rule(b);
```

DiaGen: Syntax-direct editing (cont.)

A more complicate example:





DiaGen: Syntax-direct editing (cont.)

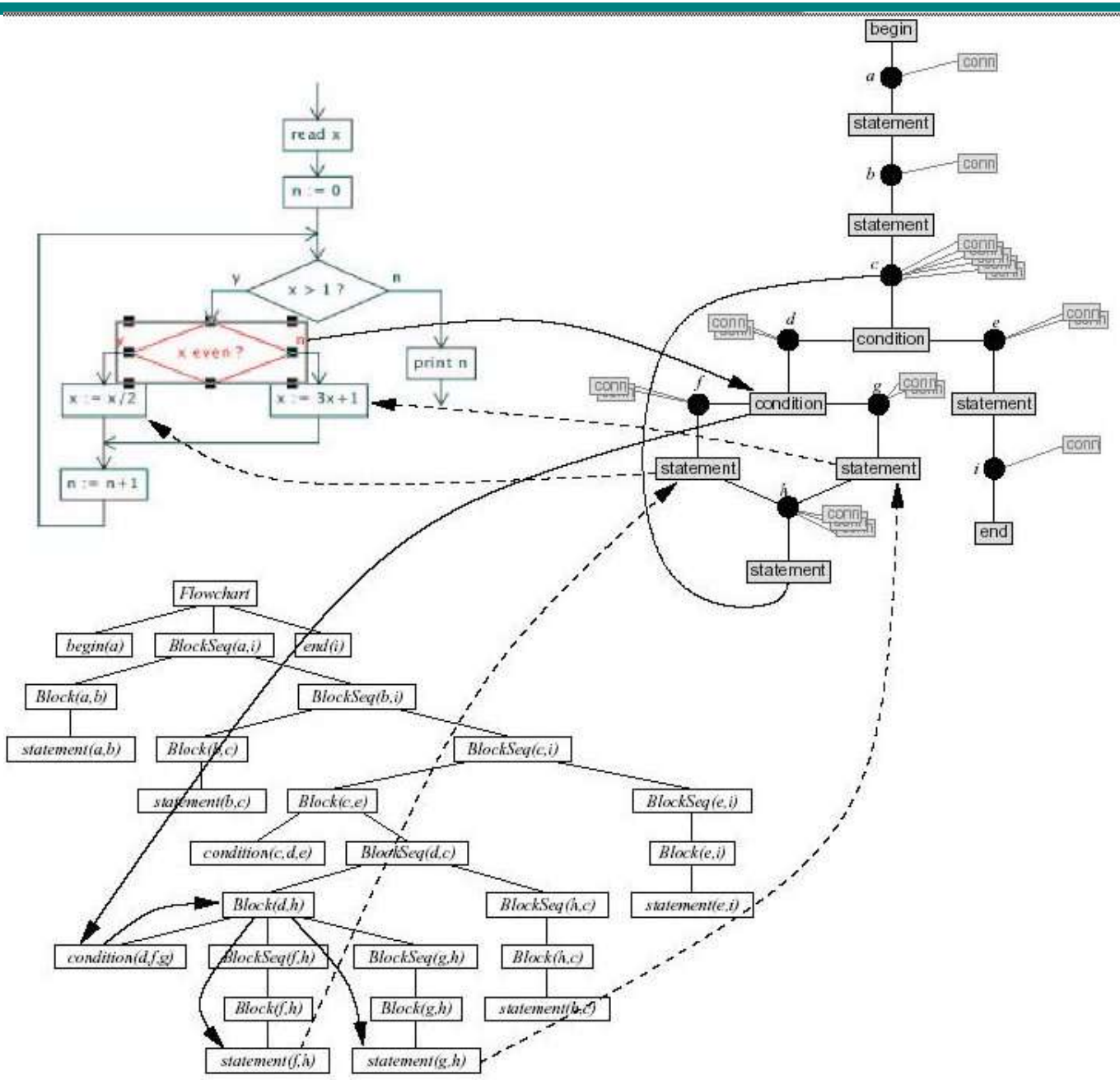
Why is it more complicated:

- The number of edges to be removed is unknown when is operation is specified
- Difficult to decide if a diagram component and its hyperedge belong to the conditional block when only considering the hypergraph model.

Solution:

Also considering the syntactic info in the derivation tree.

DiaGen: Syntax-direct editing (cont.)



DiaGen: Automatic Layout

Since DiaGen supports syntax-direct editing, automatic layout mechanism is needed.

Diagen supports two kinds of automatic layout:

- Manually program the layout module
- Constraint-based specification and computing diagram layout by a constraint solver

DiaGen: Automatic Layout (cont.)

A valid diagram layout is specified by a set of constraints on the component layout attributes. (i.e. The position).

We need to define layout constraints:

- In hypergraph grammar which is used in parsing step
- In the rule set which specifies the reduction step

Overview of DiaGen (cont.)

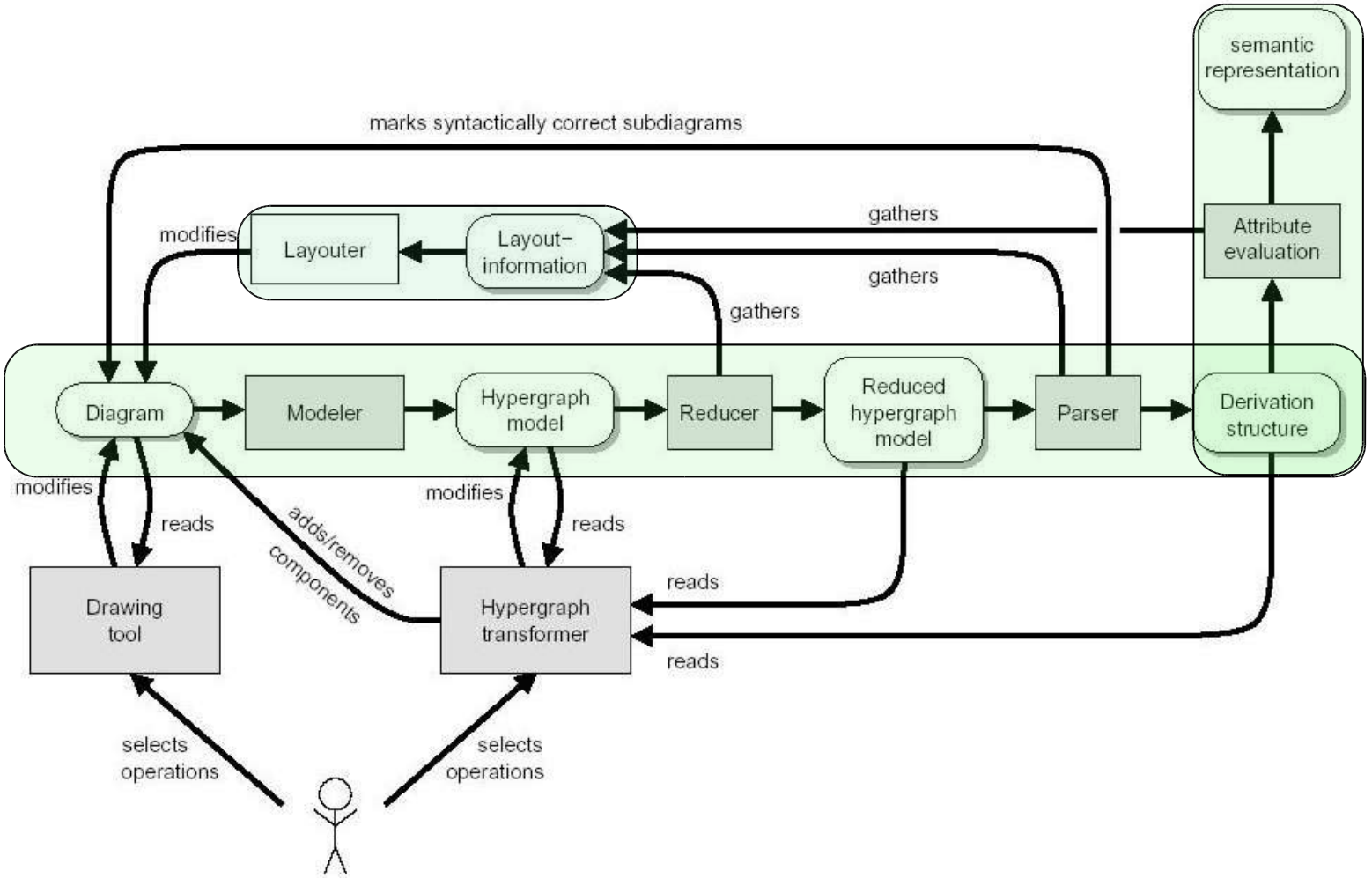


Fig. 2. Architecture of a diagram editor based on DiaGen.

Conclusion

- A rapid-prototyping tool based on hypergraph transformation
- DiaGen supports both free-hand and syntax-direct editing
- It is powerful and general.
- Syntax-direct edting is on hypergraph model, future work will study if it is sufficient to it on the reduced model.