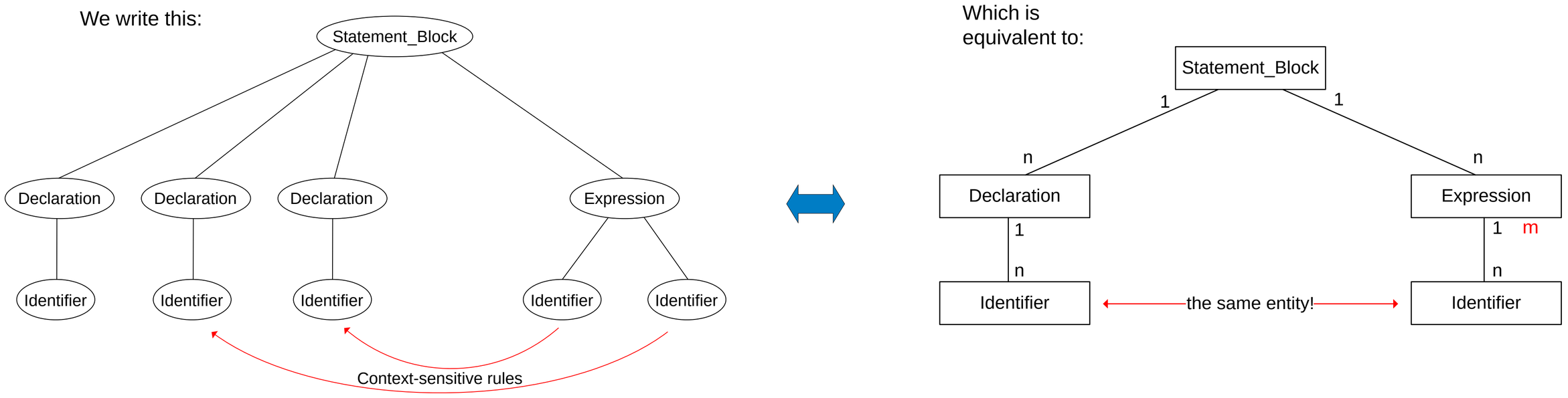
Motivation

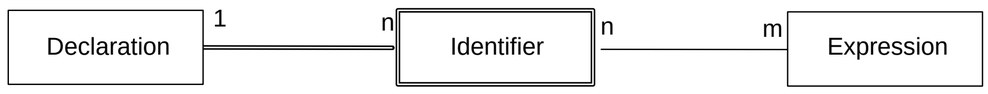
- Weak code generation techniques: Mostly template-based, hard to read and understand, difficult to maintain, code is treated as plain text.
- Lack of reuse of language-concepts (syntactic and semantic), e.g. arithmetic expressions.

What‘s the difference between models and code?

Ordinary models are represented as data (often XML or proprietary file formats) whereas languages are defined by their (context-free) grammar.



But mean that:



→ Fundamental question: Where can you find the expressiveness of data modelling techniques in the Chomsky Hierarchy?

An Approach to Model Transformation/Code Generation

The semantics of models/languages lie in what you can transform them to.

→ Generation rules in a declarative manner

→ Respecting the syntax of the target-language

→ Defining mappings between spanning trees of the source- and the target-model (graph traversal)

```
Component c → insert Java-Class(Name = c.Name, Visibility = public) jc {

   insert Constructor(Name = jc.Name);

   Port (Type == "sender")   p    → insert Method(Name = c.Name+"_"+p.Name){…};
   Port (Type == "receiver") p    → insert Method(Name = c.Name+"_"+p.Name){…},
                                    insert Method(Name = "CS"+c.Name+"_"+p.Name){…},
                                    insert Attribute(Name = "isCalled_"+p.Name, Type = boolean);
   DataElement d → GenAttr;
}

GenAttr : DataElement d → insert Attribute(Name = d.Name, Type = string);
```

Intention

Composition of Domain Specific Languages:

- Meta-modelling using a building blocks approach.
- If a set of model elements are introduced into a DSL, all transformation rules that rely on this set could be reused.
- The concept of free transformation rules build the basis for this kind of language composition.