EDAN65: Compilers, Lecture 9A

Static analysis

Görel Hedin

Revised: 2016-09-26

Program analysis

compute program properties

to transform code
to optimize code
to find bugs
to support interactive tooling
to measure quality
...

Static

on the source code
or on compiled code

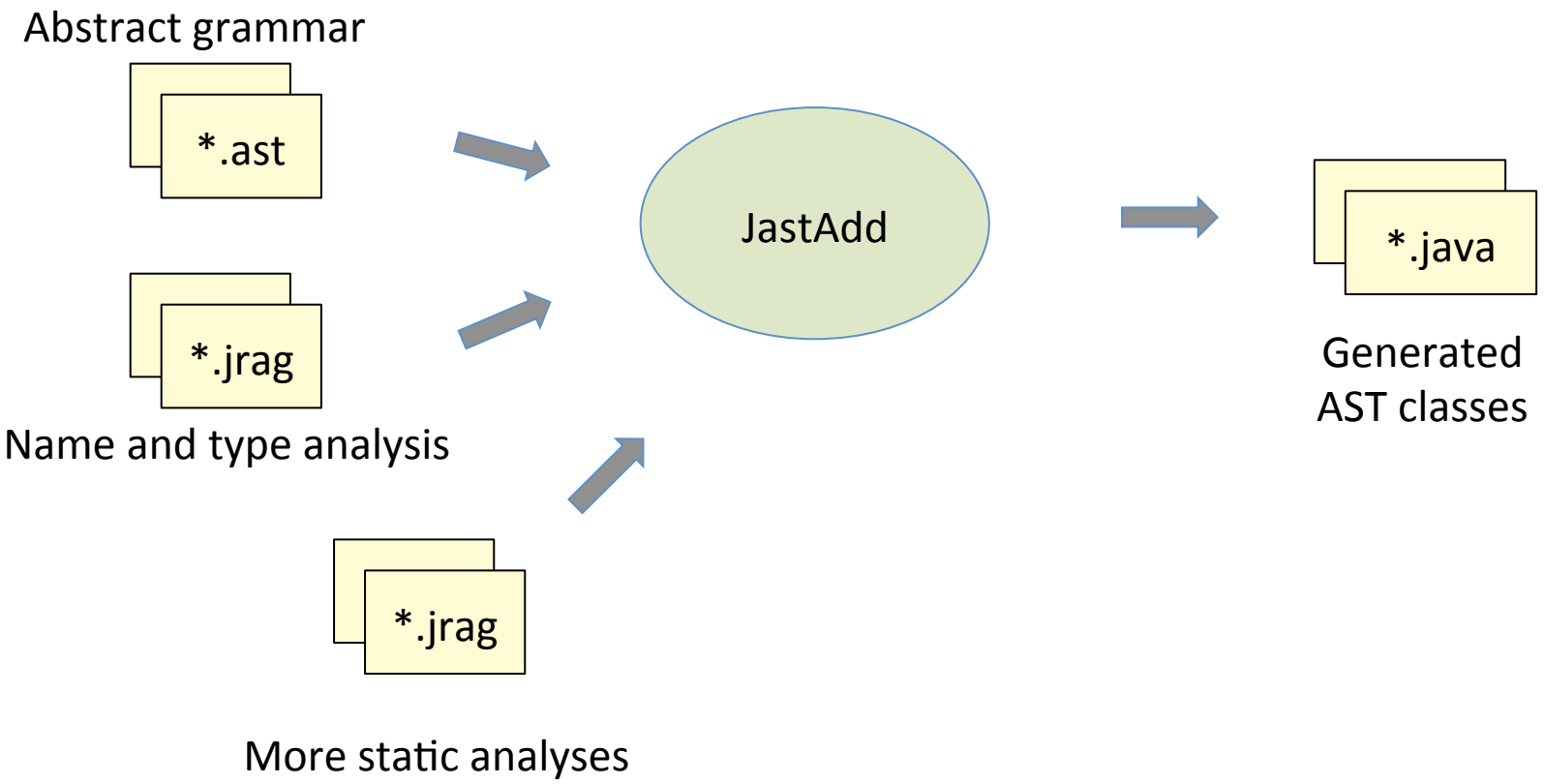
typically extends name and type analysis done in the compiler

a conservative approximation of all possible program runs

Dynamic

on a running program

Modular addition of static analyses in JastAdd



Example static analyses

name-analysis.jrag

type-analysis.jrag

control-flow.jrag

What statements can be reached from a given point?
Are there statements that are unreachable in a method?

data-flow.jrag

What statements affect the value of a given variable at a given point?
Are there statements that are unnecessary in the method?

call-graph.jrag

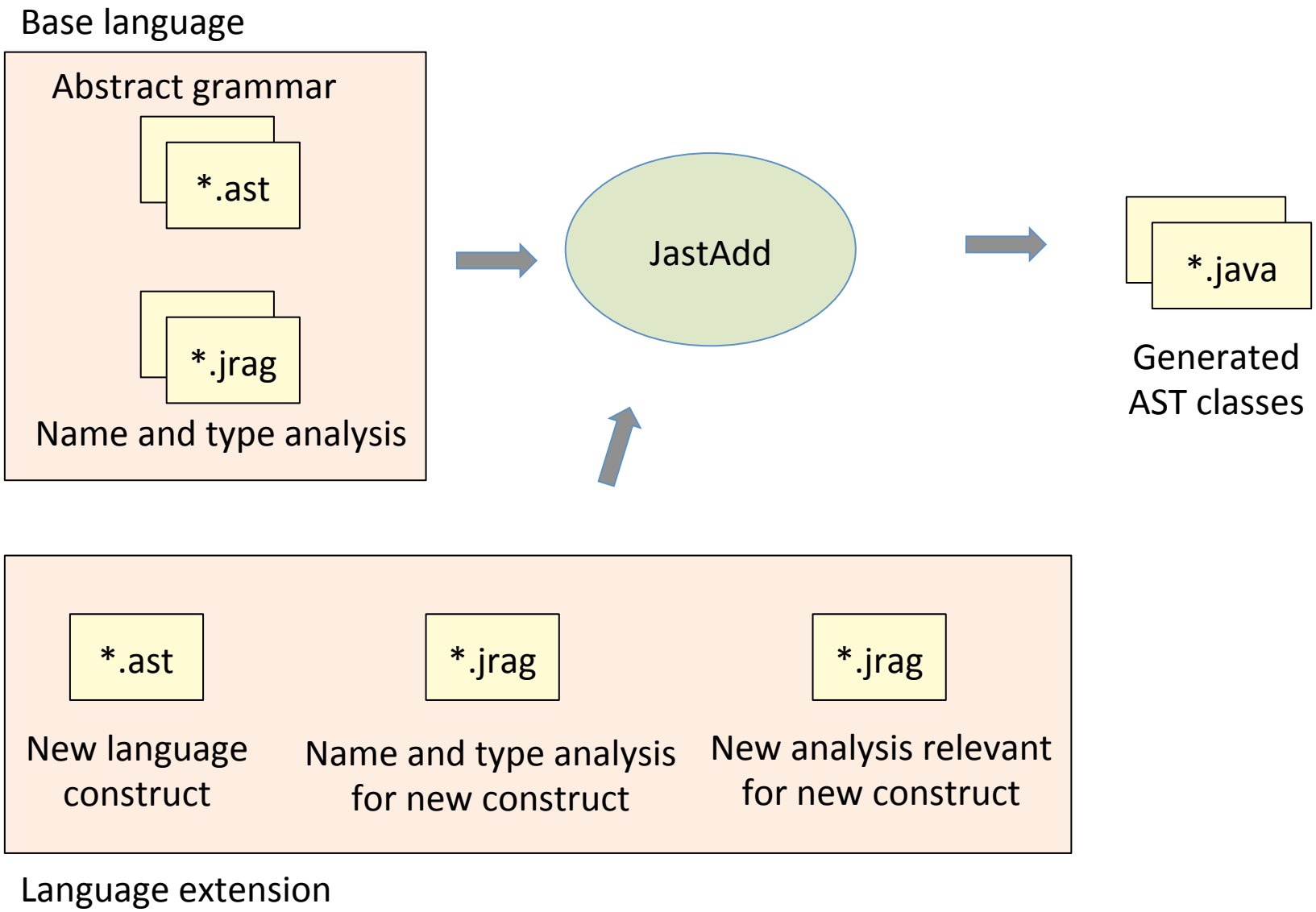
What methods are called by a given method?
Are there methods that are never called?

metrics.jrag

Compute some useful metrics of a method, class or program.

....jrag

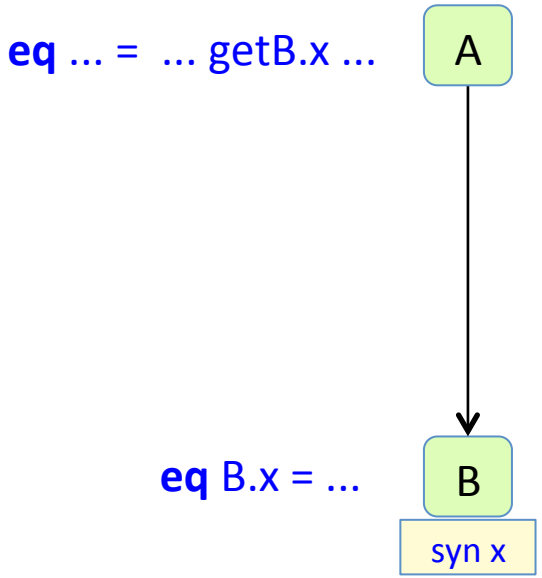
Modular language extension in JastAdd



Recall JastAdd mechanisms

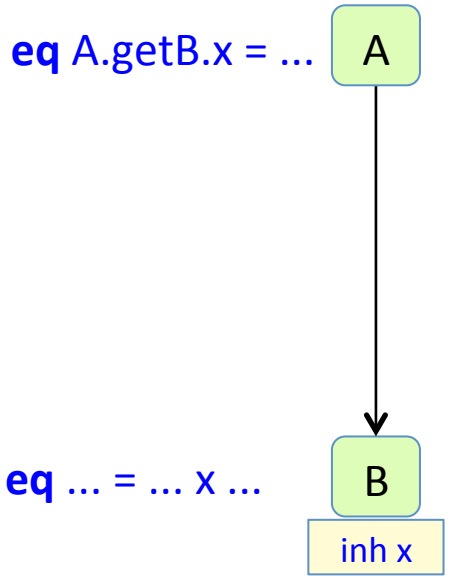
Synthesized
Inherited
Broadcasting
Reference
Parameterized
NTA
Collection
Circular

Synthesized attribute



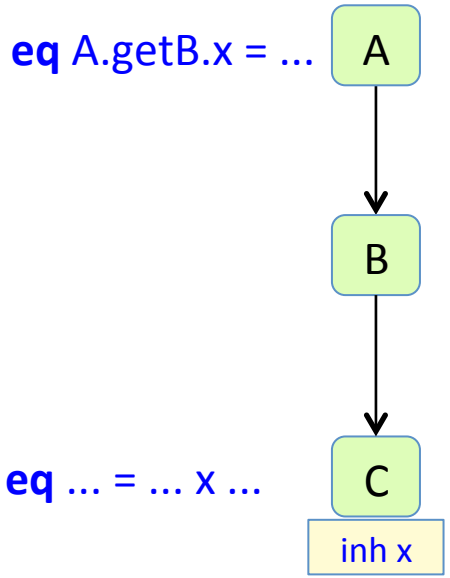
Define in the node itself. Use in parent.

Inherited attribute attribute



Use in the node itself. Define in a parent.

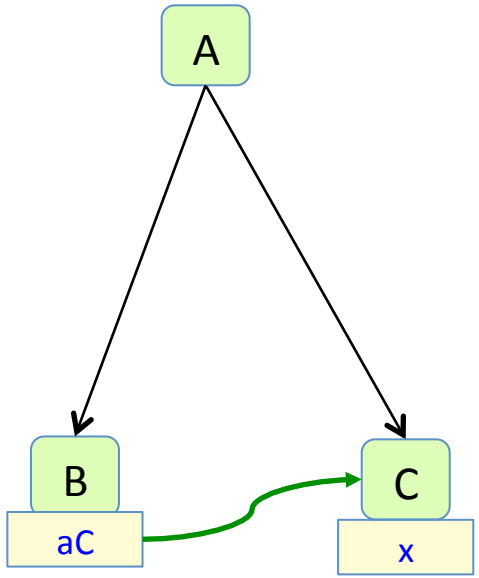
Broadcasting



The definition does not have to be in the immediate parent.

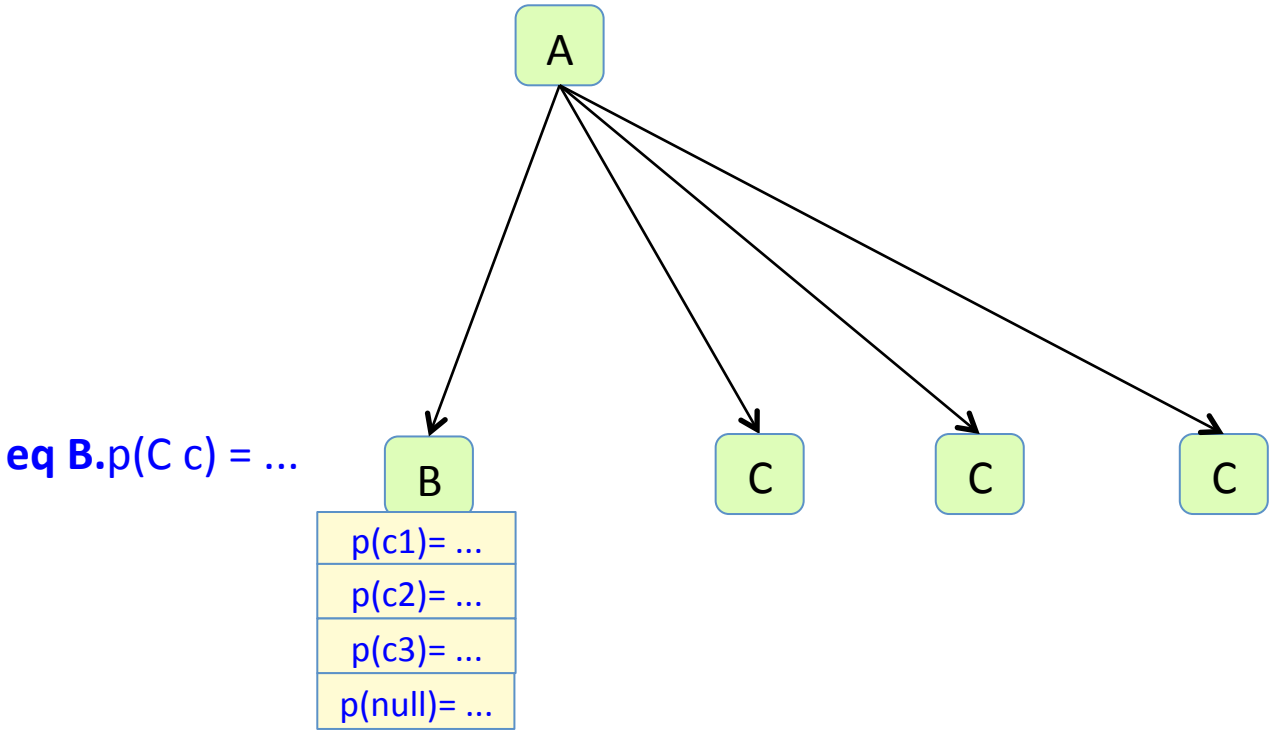
Reference attributes

eq B.aC = ...
eq ... = ... aC.x ...



An attribute can be a reference to another node.
Attributes of that node can be accessed.

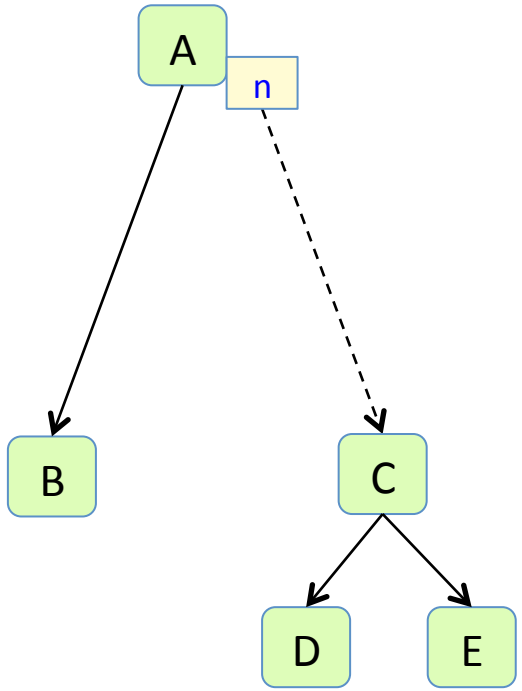
Parameterized attributes



An attribute can have parameters.
There is one attribute instance for each possible parameter combination.

Nonterminal attributes (NTAs)

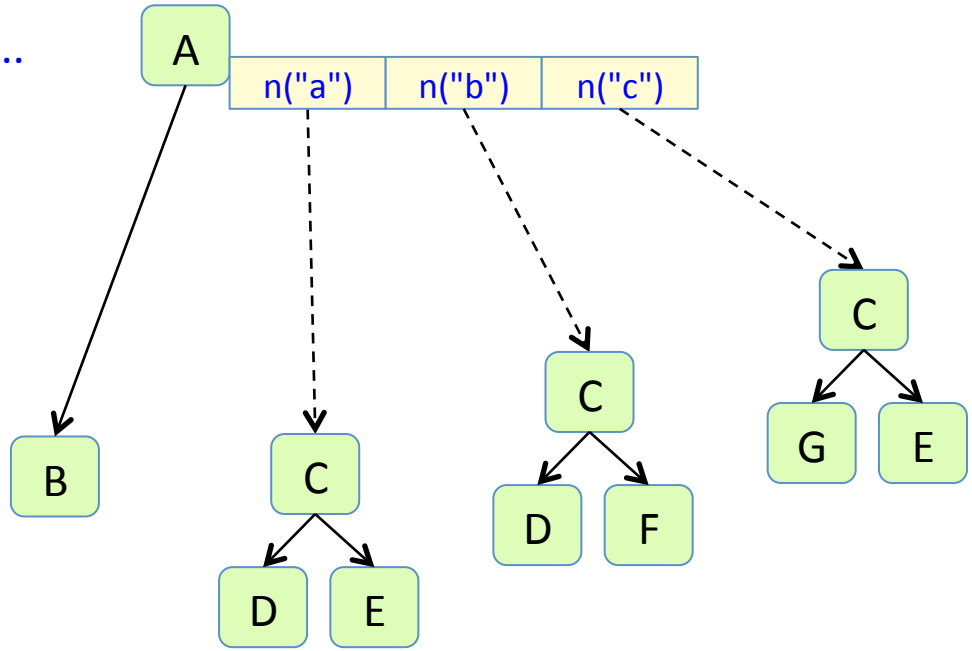
nta C A.n = ...



An attribute can be a new fresh subtree.

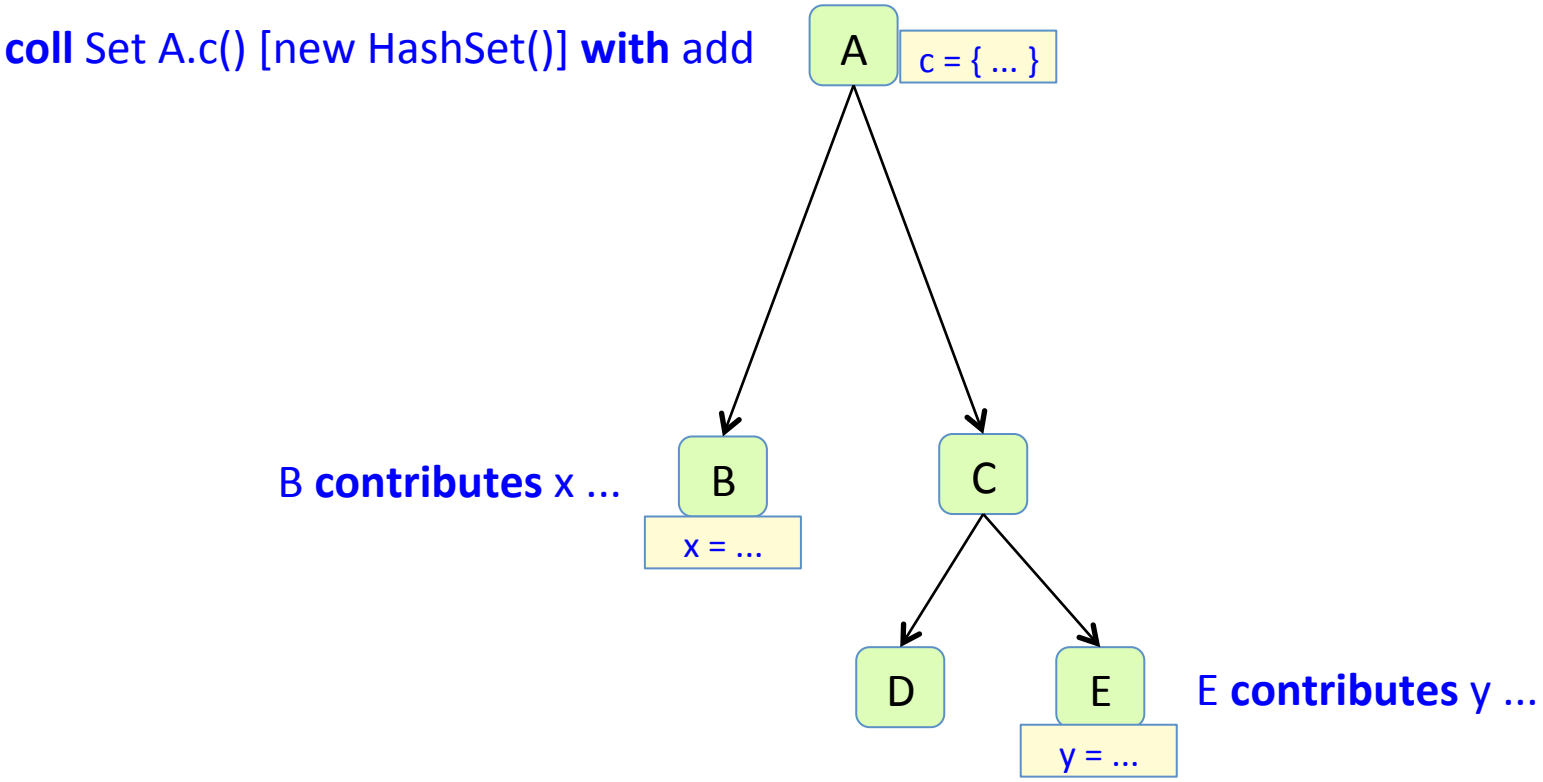
Parameterized nonterminal attribute

nta C A.n(String s) = ...



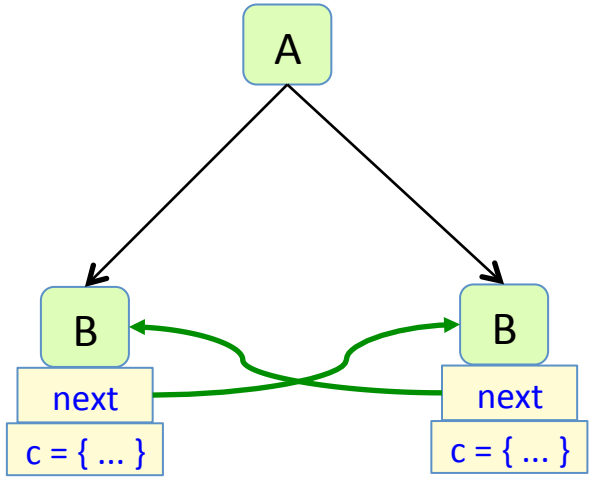
An NTA can be parameterized.

Collection attributes



A collection is a combination of contributions.

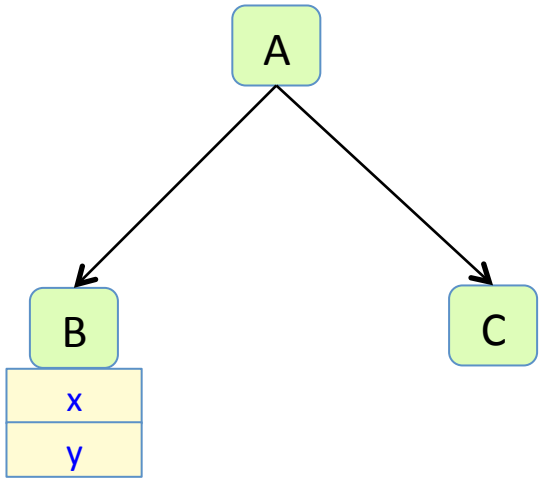
Circular attributes



syn Set B.c() **circular** [new HashSet()] = ... next().c() ...;

A circular attribute depends (transitively) on itself.
The evaluation algorithm uses fixed point iteration.

Think declaratively!



- What do you want to compute?
- What properties would allow you to easily compute that? Declare as attributes.
- Make an attribute syntheised if it depends on information in the node.
  Inherited if it only depends on context.
- When defining an attribute, make up new properties/attributes that would make it easy to compute.
- Don't think about the order of computation.