Tree Adjoining Grammars: XMG (Session 3)
Reusing fragments

Laura Kallmeyer & Simon Petitjean

HHU Düsseldorf

WS 2015

20.11.2015

Plan

1. Recap

2. Using abstractions

Recap

- First example: copy language
- two (almost) identical trees → parametrized abstraction
- Second example: intransitive, transitive and Wh-extracted trees
- Task: defining finer abstractions and combining them

Using abstractions: reminder

- in XMG, abstractions are classes
- they can be used in other abstractions by two means
  - import
  - call
- *imports* are done in the header of the class, *calls* inside its body

Using abstractions: importing or calling?

What is different?

- accessing variables: directly (extended scope) or using an operator (dot)
- using disjunction (only possible with calls)
- using parameters (only possible with calls)

What is not different?

- the constraints defined in the abstraction are added to the current description
- variables which are not exported are not accessible (only local)

Using abstractions: accessing variables

When importing:

- `import myclass[]`
- all variables exported by `myclass` are added to the scope

When calling:

- `?C=myclass[]`
- variables exported by `myclass` can be accessed with the dot operator: `?C.?N=?D.?N`

Exporting variables

```
class myClass
import myOtherClass[]
export ?S
declare ?S ?V ?N
{
  ...
}
```

- inside `myClass`, `?S, ?V, ?N` and the variables exported by `myOtherClass` can be used
- when a class imports `myClass`, `?S` and all the variables exported by `myOtherClass` are added to the local environment
- when calling `myClass`, the same set of variables is available via the dot notation

Conclusion

- Abstractions can be used in different ways
- Most of the time, several options are possible (as long as no disjunction is needed)
- If a class in not valued and not reused (imported or called), it is useless
- Now: final example and application to our small grammar