Parsing LCFRS in Vanda

Thomas Ruprecht

May 1, 2018

LCFRS [Vijay-Shanker, Weir, and Joshi 1987; Denkinger 2016]

- rules like

$$\mathrm{ROOT} \to f(\mathrm{S}, \$,, \$.)$$

where

$$f(\langle x_{0,0}, x_{0,1}\rangle, \langle x_{1,0}\rangle, \langle x_{2,0}\rangle) = \langle x_{0,0} x_{1,0} x_{0,1} x_{2,0}\rangle$$

  - composition representation implies function using string substitution
  - linear and non-deleting composition

LCFRS [Vijay-Shanker, Weir, and Joshi 1987; Denkinger 2016]

- rules like

$$\text{ROOT} \to f(\text{S}, \$,, \$.)$$

where

$$f(\langle x_{0,0}, x_{0,1}\rangle, \langle x_{1,0}\rangle, \langle x_{2,0}\rangle) = \langle x_{0,0} x_{1,0} x_{0,1} x_{2,0}\rangle$$

  - composition representation implies function using string substitution
  - linear and non-deleting composition

```
1 data PMCFG nt t   = PMCFG [nt] [Rule nt t]
2 newtype Rule nt t = Rule ((nt, [nt]), [[VarT t]])
3 data VarT t       = T t | Var Int Int
```

Ranges and instantiations

- range: pair of indices limiting subword, e.g. aa bccd

Ranges and instantiations

- range: pair of indices limiting subword, e.g. ${}_0\mathsf{aa}_2\mathsf{bccd} \to (0, 2)$

Ranges and instantiations

- range: pair of indices limiting subword, e.g. aa bccd $\to (0, 2)$
- instantiated composition: replace terminal sequences with ranges in word
  - $\langle a x_{1,1}, c x_{1,2} \rangle_{aabccd} = \{ \langle (0,1) x_{1,1}, (3,4) x_{1,2} \rangle, \langle (0,1) x_{1,1}, (4,5) x_{1,2} \rangle, \dots \}$

Ranges and instantiations

- range: pair of indices limiting subword, e.g. aa bccd $\to (0, 2)$
- instantiated composition: replace terminal sequences with ranges in word
  - $\langle a x_{1,1}, c x_{1,2} \rangle_{aabccd} = \{ \langle (0,1) x_{1,1}, (3,4) x_{1,2} \rangle, \langle (0,1) x_{1,1}, (4,5) x_{1,2} \rangle, \dots \}$

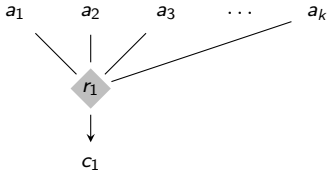
Ranges and instantiations

- range: pair of indices limiting subword, e.g. aa bccd $\rightarrow (0, 2)$
- instantiated composition: replace terminal sequences with ranges in word
  - $\langle ax_{1,1}, cx_{1,2} \rangle_{aabccd} = \{\langle (0,1)x_{1,1}, (3,4)x_{1,2} \rangle, \langle (0,1)x_{1,1}, (4,5)x_{1,2} \rangle, \dots\}$

```
1  instantiate :: [t]
2              → Function t
3              → [InstantiatedFunction]
4  instantiate w'
5    = mapM (mapMaybe concVarRange ∘ sequence ∘ instantiateComponent w')
6    where
7      instantiateComponent :: [t] → [VarT t] → [[VarT Range]]
8      instantiateComponent _ [] = [[ T Epsilon ]]
9      instantiateComponent w fs = map (instantiateCharacter w) fs
10
11     instantiateCharacter :: [t] → VarT t → [VarT Range]
12     instantiateCharacter _ (Var i j)  = [Var i j]
13     instantiateCharacter w (T c)      = map T $ singletons c w
```

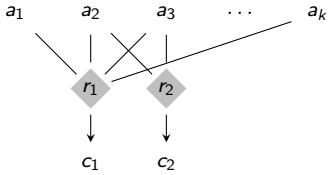
Weighted deduction [Shieber, Schabes, and Pereira 1995]

- rule application



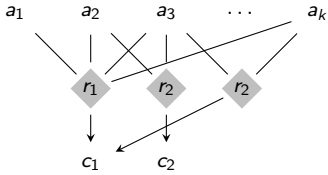
Weighted deduction [Shieber, Schabes, and Pereira 1995]

- rule application



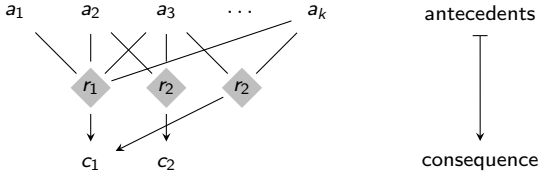
Weighted deduction [Shieber, Schabes, and Pereira 1995]

- rule application



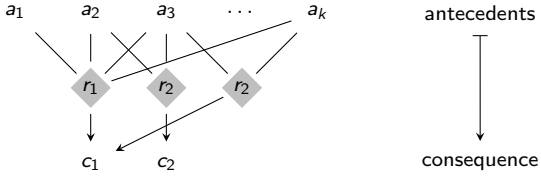
Weighted deduction [Shieber, Schabes, and Pereira 1995]

- rule application



Weighted deduction [Shieber, Schabes, and Pereira 1995]

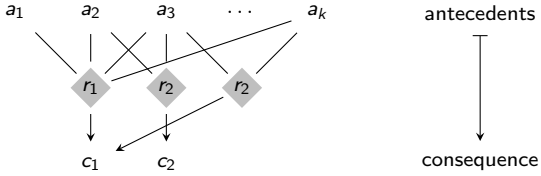
- rule application



- enumerate items using Knuth's algorithm [Nederhof 2003]

Weighted deduction [Shieber, Schabes, and Pereira 1995]

- rule application

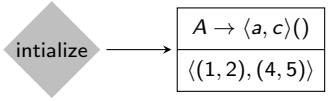


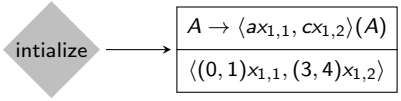
- enumerate items using Knuth's algorithm [Nederhof 2003]

```haskell
-- with: * it - items
--       * wt - weights
--       * ct - container
type DeductionRule it wt ct = Either [(it, wt)]
                                     (it → ct → [(it, wt)])
```

Example: CYK parsing [Seki et al. 1991]

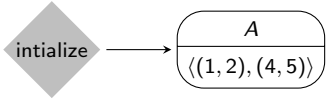
- *initialize*

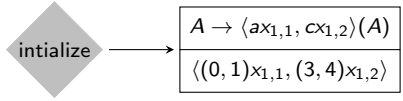




Example: CYK parsing [Seki et al. 1991]

- *initialize*

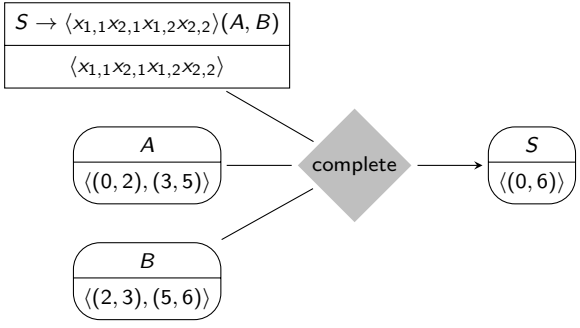




Example: CYK parsing [Seki et al. 1991]

- *initialize*

- *complete*



Example: CYK parsing [Seki et al. 1991], continued

```
data Item nt t wt = Active (Rule nt t) wt InstantiatedFunction
                  | Passive nt Rangevector (Backtrace nt t wt) wt
data Backtrace nt t wt = Backtrace (Rule nt t) wt [Rangevector]
```

Example: CYK parsing [Seki et al. 1991], continued

```
data Item nt t wt = Active (Rule nt t) wt InstantiatedFunction
                  | Passive nt Rangevector (Backtrace nt t wt) wt
data Backtrace nt t wt = Backtrace (Rule nt t) wt [Rangevector]
```

```
prediction :: [t]
           → [(Rule nt t, wt)]
           → Map.HashMap nt (wt, wt)
           → DeductionRule (Item nt t wt) wt (Container nt t wt)
prediction word rules ios
  = Left
  $ catMaybes
  [ implicitConversion (Active r w fw, i <.> o)
  | (r@(Rule ((a, as), f)), w) ← rules
  , fw ← instantiate word f
  , let i = w <.> foldl (<.>) one (map (fst ∘ (ios Map.!)) as)
        o = snd $ ios Map.! a
  ]
```

Example: CYK parsing [Seki et al. 1991], continued

```
completion :: Map.HashMap nt (wt, wt)
           → DeductionRule (Item nt t wt) wt (Container nt t wt)
completion ios = Right app
  where
    app item@(Active (Rule ((_, as), _)) _ _) (ps, _, _)
      = [ consequence
        | pas ← mapM (lookupWith Passive ps) as
        , consequence ← consequences (item:pas)
        ]
```