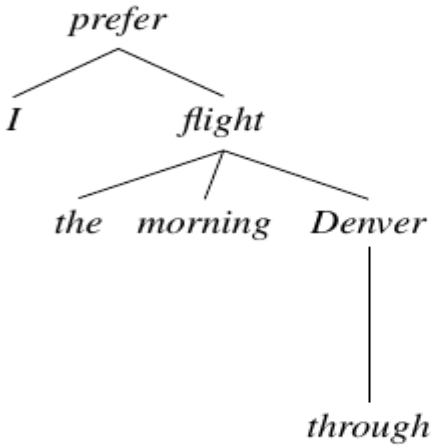
Dependencia parszolás

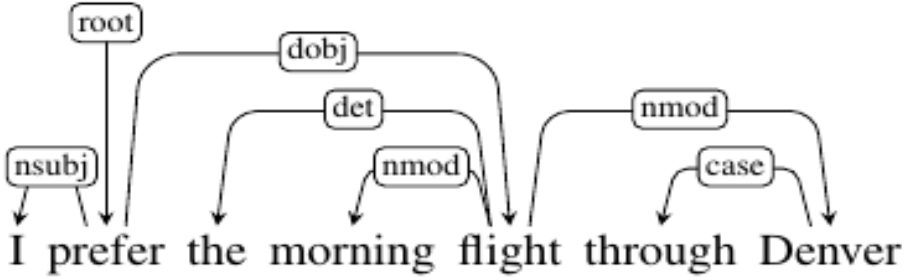
Lévai Dániel

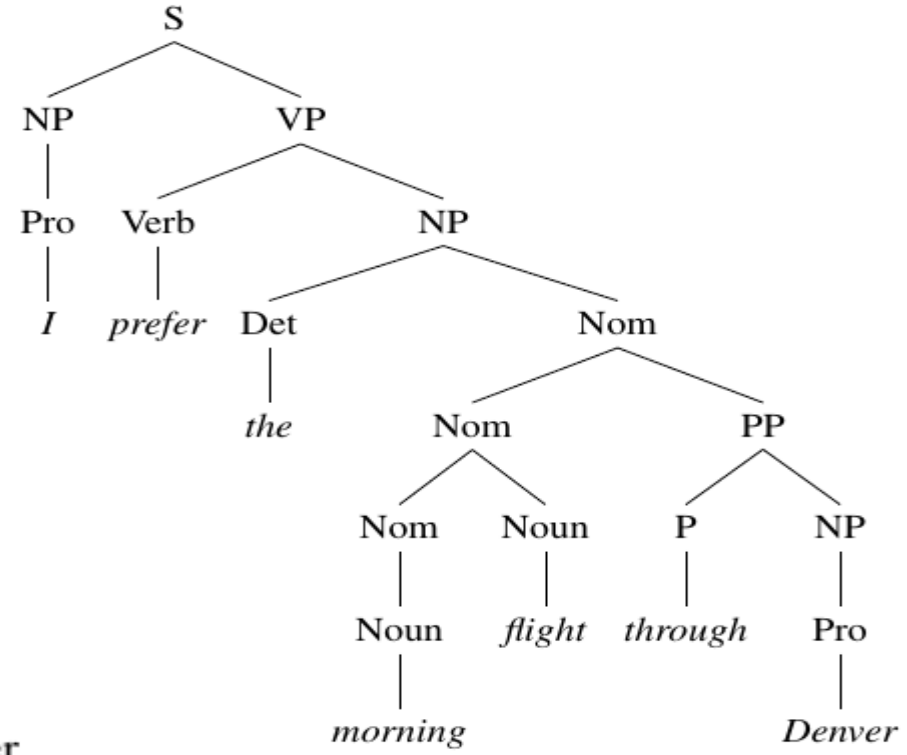
Dependencia vs. PSG, 1.

- **Két fő nyelvtani formalizmus van:**
  - Phrase structure grammar (HPSG, LFG, Chomsky, CF)
    - Vannak csúcsok (frázisok, szavak), élek, de több csúcs van, mint szó a mondatban
  - Dependencia:
    - Annyi csúcs van, amennyi szó, az élek irányítottak és nevük van
    - Jobban kezeli a szabad szórendet

Dependencia vs. PSG, 2.







Dependencia relációk

- Kétargumentumú relációk: fej és dependens
- De lehet három argumentumú is:
  - Fej
  - Dependens
  - Nyelvtani funkció
- UD

Clausal Argument Relations	Description
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

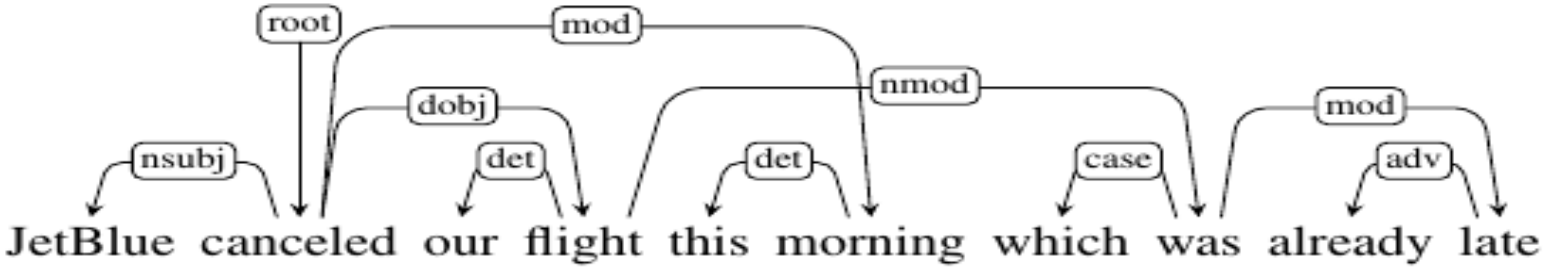
Figure 15.2 Selected dependency relations from the Universal Dependency set. (de Marneffe et al., 2014)

Dependencia formalizmusok

- **A dependencia struktúra irányított gráf**
  G = (V, A), ahol az élek rendezett párok (hármasok)
- **Van gyökér (root), körmentes (acyclic), azaz:**
  - Root: nincsen bemenő, csak kimenő éle
  - A gyökéren kívüli csúcsoknak pontosan 1 bemenő éle van
  - Minden csúcsból van pontosan egy út a gyökérbe

Projektivitás

- Egy él projektív, ha az él mentén lévő szavakhoz el lehet jutni a kezdőcsúcsból



- Flight → was nem projektív
- De a legtöbb adathalmaz konstituens fákból lett létrehozva, azok pedig mindig projektívek

Dependencia fabankok

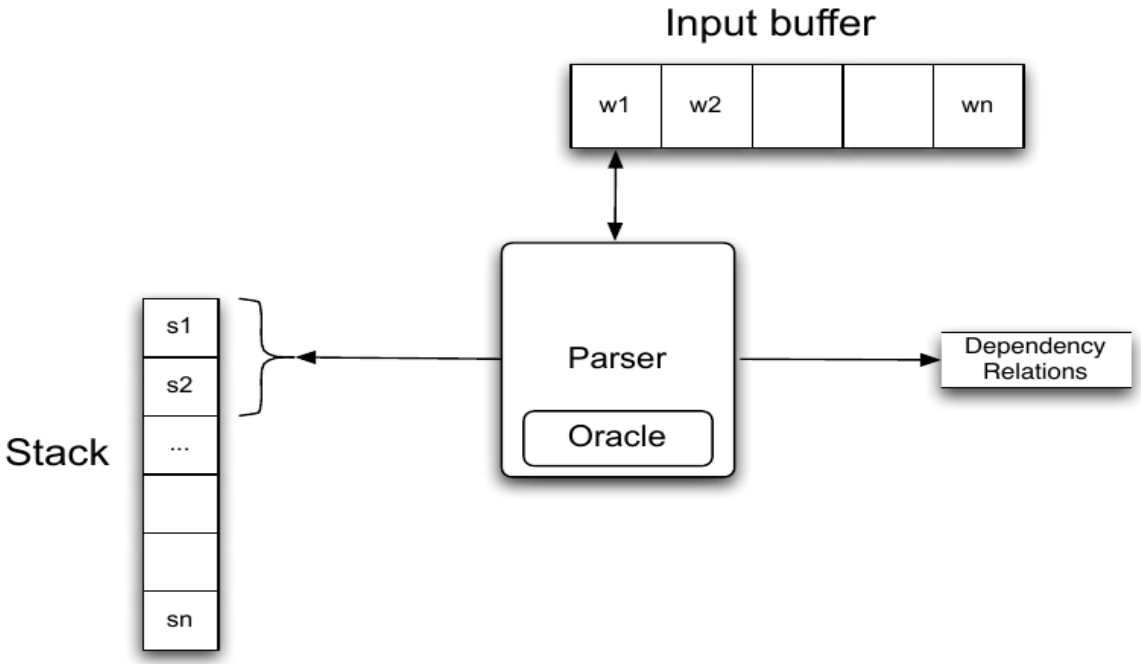
- Konstituensfákból hozzák létre
- Csak angolra szokták használni
- Idegennyelvekre pedig kézzel annotálnak
- A fejeket ki lehet nyerni a konstituens-fákból, és abból lehet építeni dependenciafát

Parszolás

1. **Dinamikus programozás**
   - Mint a CKY, de lassú O(n3)
2. **Gráf algoritmusok (e-magyar)**
3. **Constraint satisfaction**
4. **Determinisztikus parszolás**

Mohó algoritmus, 1.

- Stack, buffer, orákulum
- 3 fajta lépés:
  - LeftArc
  - RightArc
  - Shift
- Orákulum
- Megkötések



Mohó algoritmus, 2.

- **Mohó algoritmus, mert:**
  - Lokális döntéseket hoz
  - Lineáris

function DEPENDENCYPARSE(*words*) **returns** dependency tree

state ← {[root], [*words*], [] } ; initial configuration
while *state* **not final**
  t ← ORACLE(*state*) ; choose a transition operator to apply
  state ← APPLY(*t*, *state*) ; apply it, creating a new state
return *state*

Orákulum, 1

- **Egy adott bemeneti konfigurációhoz jó kimenetet kell társítani**
  - Azaz jó lépést
- **Gold adatból**

Orákulum, 2

- **LeftArc legyen, ha:**
  - Jó fej-dependens relációt hoz létre
- **RightArc legyen, ha:**
  - Jó fej-dependens relációt hoz létre ÉS
  - A fölötte lévő szó minden dependens reláció megvan már (mert ezzel a lépéssel ürítünk a stacken)
- **Shift, amúgy**

Orákulum, 3

- **Bemeneti feature lehet:**
  - Morfoszintaktikai jegyek
  - Lexikális jegyek (the…)
  - Ezek kombinációja
- **SVM, logreg, NN, …**

Arc eager transition system, 1

- **LeftArc**
  - Fej-dependens reláció az input buffer eleje és a stack teteje között, majd a stack popolása
- **RightArc**
  - Fej-dependens a stack teteje → input buffer eleje, majd Shift
- **Shift**
- **Reduce**

Arc eager transition system, 2

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, the, flight, through, houston]	RIGHTARC	(root → book)
1	[root, book]	[the, flight, through, houston]	SHIFT	
2	[root, book, the]	[flight, through, houston]	LEFTARC	(the ← flight)
3	[root, book]	[flight, through, houston]	RIGHTARC	(book → flight)
4	[root, book, flight]	[through, houston]	SHIFT	
5	[root, book, flight, through]	[houston]	LEFTARC	(through ← houston)
6	[root, book, flight]	[houston]	RIGHTARC	(flight → houston)
7	[root, book, flight, houston]	[]	REDUCE	
8	[root, book, flight]	[]	REDUCE	
9	[root, book]	[]	REDUCE	
10	[root]	[]	Done	

Step	Stack	Word List	Predicted Action
0	[root]	[book, the, flight, through, houston]	SHIFT
1	[root, book]	[the, flight, through, houston]	SHIFT
2	[root, book, the]	[flight, through, houston]	SHIFT
3	[root, book, the, flight]	[through, houston]	LEFTARC
4	[root, book, flight]	[through, houston]	SHIFT
5	[root, book, flight, through]	[houston]	SHIFT
6	[root, book, flight, through, houston]	[]	LEFTARC
7	[root, book, flight, houston]	[]	RIGHTARC
8	[root, book, flight]	[]	RIGHTARC
9	[root, book]	[]	RIGHTARC
10	[root]	[]	Done

Más átmenet alapú módszerek

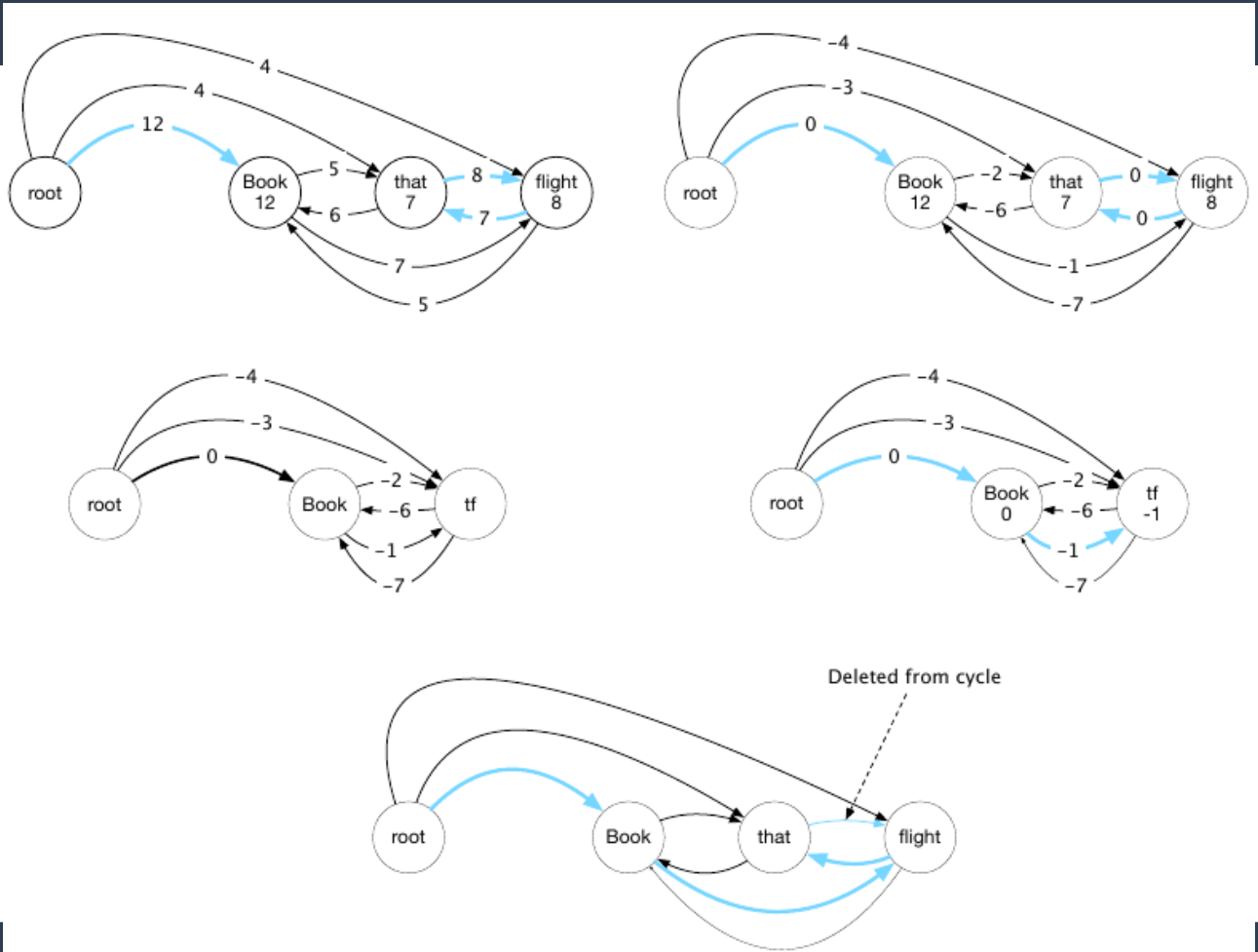
- **Vannak**
- **Fontos technikája a beam search:**
  - Minden ponton minden lépést megteszünk
  - Majd ezeket pontozzuk
  - És csak a jobbakat tartjuk meg

Gráfalapú módszerek

- A gold adaton van egy fánk (gráfunk)
- Szeretnénk úgy fákat keresni, hogy maximalizálják a score-t
- A score pedig az élek score-jának összege
- Azért van ezekre szükség, mert az átmeneten alapuló módszerek rosszul kezelik a hosszú dependenciákat (és a nem projektív éleket is)

$$\hat{T}(S) = \underset{t \in \mathcal{G}_S}{\operatorname{argmax}}\, score(t, S) \qquad score(t, S) = \sum_{e \in t} score(e)$$

Maximális irányított feszítőfa



Kiértékelés

- Labeled/unlabeled accuracy score (LAS, UAS)
- Adott szóhoz jó-e a fej/adott szóhoz jó-e a fej és a reláció
- Vagy adott típusú dependenciákat nézve accuracy, recall, precision, F-score

Összefoglalás

- Szavak bináris relációjával írjuk le egy mondat struktúráját
- A reláció fej-dependens-típus
- Nagyon hasznos információkinyerésre
- Sokféle algoritmus van, a mohó stack-alapú és maximális feszítőfa algoritmusokat néztük meg
- Evaluálni viszonylag egyszerű

Köszönöm a figyelmet!