

# Language and Statistics II

---

Lecture 5: Dynamic programming algorithms as grounded weighted logic programs, continued

# Outline for Today

---

- Analyzing program complexity
- Program transformations for runtime efficiency
- A general algorithm for solving semiring-weighted logic programs bottom-up
- Properties of problems suitable for dynamic programming

# Static Analysis (McAllester, 2002)

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

# Space Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

Assume every theorem that can be proved, is.

Assume every proved theorem needs to be stored.

Assume (pessimistically) a dense grammar.

$g$  = number of nonterminals,  $t$  = number of terminals

$n$  = length of the sentence

# Space Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

# Space Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

word(W, I): n values of I, 1 word per position ...  $O(n)$

# Space Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

$\underline{\text{word}}(W, I)$ :  $n$  values of  $I$ , 1 word per position ...  $O(n)$

$\underline{\text{length}}(N)$ :  $O(1)$

# Space Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

$\underline{\text{word}}(W, I):$   $n$  values of  $I$ , 1 word per position ...  $O(n)$

$\underline{\text{length}}(N):$   $O(1)$

$\underline{\text{unary}}(X, W):$   $O(g^t)$

$\underline{\text{binary}}(X, Y, Z):$   $O(g^3)$

# Space Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

$\underline{\text{word}}(W, I):$   $n$  values of  $I$ , 1 word per position ...  $O(n)$

$\underline{\text{length}}(N):$   $O(1)$

$\underline{\text{unary}}(X, W):$   $O(g^t)$

$\underline{\text{binary}}(X, Y, Z):$   $O(g^3)$

$\underline{\text{constit}}(X, I, K):$   $O(gn^2)$

# Classic (Chart) View

---

*“to”*

1	2	3	4	5	6	7	
							1
							2
							3
							4
							5
							6
							7

*“from”*

each cell holds a map from nonterminals to values, plus backpointers

# Runtime Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

Assume every theorem that can be proved, is, every way that it can be proved, and constant time per inference.

Assume (pessimistically) a dense grammar.

$g$  = number of nonterminals,  $t$  = number of terminals

$n$  = length of the sentence

# Runtime Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

X can take g values

I can take n values

W can take 1 value (given I)

# Runtime Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

X can take g values

I can take n values

W can take 1 value (given I)

...  $O(gn)$

# Runtime Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

N can take 1 value

# Runtime Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

N can take 1 value

...  $O(1)$

# Runtime Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

X, Y, Z can take g values each

I, J, K can take n values each

# Runtime Analysis

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

X, Y, Z can take g values each

I, J, K can take n values each

...  $O(n^3g^3)$

# Program Transformations

---

- Generally speaking, we can define mechanical transformations on programs.
- Some transformations are **semantics-preserving**.  
(Obviously some aren't.)
- Today we'll talk about a semantics-preserving transformation called “binarizing” or “folding.”

# CKY, Transformed

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \left( \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K) \right) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

Same semantics (i.e., same values for all theorems):

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{foo}(Y, Z, I, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{foo}(Y, Z, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

# CKY, Transformed

---

Space requirements:  $O(gn^2)$  before, now  $O(g^2n^2)$

Same semantics (i.e., same values for all theorems):

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{foo}(Y, Z, I, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{foo}(Y, Z, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

# CKY, Transformed

---

Space requirements:  $O(g^3 + gn^2)$  before, now  $O(g^3 + g^2n^2)$

Runtime requirements:  $O(g^3n^3)$  before, now  $O(g^2n^3 + g^3n^2)$

Same semantics (i.e., same values for all theorems):

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{foo}(Y, Z, I, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{foo}(Y, Z, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{constit}(Z, J+1, K).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

# CKY, Transformed (II)

---

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \left( \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z) \right).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

Same semantics (i.e., same values for all theorems):

$\text{constit}(X, I, I) \oplus = \underline{\text{word}}(W, I) \otimes \underline{\text{unary}}(X, W).$

$\text{constit}(X, I, K) \oplus = \text{constit}(Y, I, J) \otimes \text{bar}(X, Y, J+1, K).$

$\text{bar}(X, Y, J+1, K) \oplus = \text{constit}(Z, J+1, K) \otimes \underline{\text{binary}}(X, Y, Z).$

$\text{goal} \oplus = \text{constit}(\text{"S"}, 1, N) \otimes \underline{\text{length}}(N).$

# Comments

---

- We've been assuming that there's some kind of temporal process in which we prove theorems **bottom-up**.
  - Consequent's value gets calculated *after* its antecedents.
- In fact, that's how most dynamic programming implementations work!
- Next up: a generic algorithm to solve semiring-weighted logic programs.

# One Caveat

---

$\text{foo}(Y, Z) \oplus = \text{bar}(W, X, Y, Z) \otimes \text{baz}(V, X, Y, Z).$

- It's okay to have variables in the antecedents that don't get mentioned in the consequent. They don't even have to "match."
- It's **not** okay to have variables in the consequent that don't match anything in the antecedents!

$\text{foo}(U, Y, Z) \oplus = \text{bar}(W, X, Y, Z) \otimes \text{baz}(V, X, Y, Z).$

- In logic programming lingo, we call this property "**fully grounded.**" (Prolog doesn't require it, but we do.)

# Algorithms

---

- Goodman (1999) put semiring weights into logic programs and suggested building the whole proof structure, then calculating the values as a second pass.
  - Doesn't make sense if we really only want one proof!
- The agenda algorithm we'll discuss constructs the proof(s) bottom-up while calculating values at the same time.

# Agenda Dynamic Programming

---

- Express the problem as a semiring-weighted logic program.
- Imagine that every possible item has a value, and our goal is to calculate it. (We are “solving a system of equations.”)
- Initially, all values take the default value of (semiring) **0**.
- We proceed by calculating **updates** to items.
  - Each update will spawn other updates.
- We use an **agenda** to keep track of the updates remaining.

# Agenda Example (Whiteboard)

---

$\text{align}(I, J) \text{ min} = \text{align}(I - 1, J - 1) + \underline{x}(X, I) + \underline{y}(X, J).$

$\text{align}(I, J) \text{ min} = \text{align}(I - 1, J - 1) + \underline{\text{subcost}} + \underline{x}(X, I) + \underline{y}(Y, J).$

$\text{align}(I, J) \text{ min} = \text{align}(I, J - 1) + \underline{\text{inscost}} + \underline{y}(Y, J).$

$\text{align}(I, J) \text{ min} = \text{align}(I - 1, J) + \underline{\text{delcost}} + \underline{x}(X, I).$

$\text{goal min} = \text{align}(M, N) + \underline{\text{xlen}}(M) + \underline{\text{ylen}}(N).$

$\text{align}(0, 0) := 0.$

# Important Issues

---

- The chart is a data structure. Make it efficient!
  - Do we need to keep everything around, forever?
- Do we need to run until the agenda is empty?
- How to order the updates?
  - Want to avoid “repropagation.”
  - Problem-specific tricks: Levenshtein? CKY?
  - Depth first, breadth first, ...
  - Best first/“uniform cost”: order by update value

# Priorities in the Agenda

---

- In general, if we want to prioritize the updates, we have to **sort** them.
- Cost of sorting?
  - Runtime increases by a factor of  $\log(\# \text{ updates})$  ...

# Some Other Points

---

- Other agenda-ordering techniques exist and can speed up processing a lot for min and max semirings.
  - “Figures of merit” (Charniak et al., 1998)
  - Generalized  $A^*$  (Klein and Manning, 2003)
- **Pruning** is another widely-used technique - but you lose the optimality guarantee.
- For counting and summing semirings, where every proof has a say in the value of “goal,” life is harder.
  - Memoization often works well, but beware of cycles.

# When to use dynamic programming?

---

- Shared substructure (each theorem gets reused a lot)
- Low-dimensional “state” space (theorems)
- Global optimality “factors” into local optimality