Choosing Word Occurrences for the Smallest Grammar Problem

Rafael Carrascosa[1], Matthias Gallé[2],
François Coste[2], Gabriel Infante-Lopez[1]

[1]NLP Group
U. N. de Córdoba
Argentina

[2]Symbiose Project
IRISA / INRIA
France

LATA
May, 25th 2010

Smallest Grammar Problem

Problem Definition

Given a sequence s, find a context-free grammar $G(s)$ of minimal size that generates exactly this and only this sequence.

Smallest Grammar Problem

Problem Definition

Given a sequence s, find a context-free grammar $G(s)$ of minimal size that generates exactly this and only this sequence.

Example

$s =$ "`how much wood would a woodchuck chuck if a woodchuck could chuck wood?`", a possible $G(s)$ (not necessarily minimal) is

$$
\begin{aligned}
S &\rightarrow \text{how much } N_2 \text{ w}N_3 \; N_4 \; N_1 \text{ if } N_4 \text{ c}N_3 \; N_1 \; N_2 \text{ ?} \\
N_1 &\rightarrow \text{chuck} \\
N_2 &\rightarrow \text{wood} \\
N_3 &\rightarrow \text{ould} \\
N_4 &\rightarrow \text{a } N_2 N_1
\end{aligned}
$$

Smallest Grammar Problem

Problem Definition

Given a sequence s, find a context-free grammar $G(s)$ of minimal size that generates exactly this and only this sequence.

Applications

- Data Compression
- Sequence Complexity
- Structure Discovery

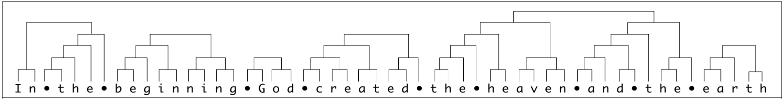
Smallest Grammar Problem

Problem Definition

Given a sequence s, find a context-free grammar $G(s)$ of minimal size that generates exactly this and only this sequence.

Applications

- Data Compression
- Sequence Complexity
- Structure Discovery



Smallest Grammar Problem

Problem Definition

Given a sequence s, find a context-free grammar $G(s)$ of minimal size that generates exactly this and only this sequence.

Remark

Not only S, but any non-terminal of the grammar generates only one sequence of terminal symbols: $cons :: \mathcal{N} \rightarrow \Sigma^*$

Smallest Grammar Problem

Problem Definition

Given a sequence s, find a context-free grammar $G(s)$ of minimal size that generates exactly this and only this sequence.

Remark

Not only S, but any non-terminal of the grammar generates only one sequence of terminal symbols: $cons :: \mathcal{N} \rightarrow \Sigma^*$

$$
\begin{array}{lcl}
S & \rightarrow & \text{how much } N_2 \text{ w}N_3 \ N_4 \ N_1 \text{ if } N_4 \text{ c}N_3 \ N_1 \ N_2 \text{ ?} \\
N_1 & \rightarrow & \text{chuck} \\
N_2 & \rightarrow & \text{wood} \\
N_3 & \rightarrow & \text{ould} \\
N_4 & \rightarrow & \text{a } N_2 N_1
\end{array}
\quad \Rightarrow \quad
\begin{array}{lcl}
\text{cons}(S) & = & s \\
\text{cons}(N_1) & = & \text{chuck} \\
\text{cons}(N_2) & = & \text{wood} \\
\text{cons}(N_3) & = & \text{ould} \\
\text{cons}(N_4) & = & \text{a woodchuck}
\end{array}
$$

Smallest Grammar Problem

Problem Definition

Given a sequence s, find a context-free grammar $G(s)$ of minimal size that generates exactly this and only this sequence.

Size of a Grammar

$$|G| = \sum_{N \to \omega \in \mathcal{P}} (|\omega| + 1)$$

Smallest Grammar Problem

Problem Definition

Given a sequence s, find a context-free grammar $G(s)$ of minimal size that generates exactly this and only this sequence.

Size of a Grammar

$$|G| = \sum_{N \rightarrow \omega \in \mathcal{P}} (|\omega| + 1)$$

$$\begin{array}{lcl}
S & \rightarrow & \text{how much } N_2 \text{ w}N_3 \ N_4 \ N_1 \text{ if } N_4 \text{ c}N_3 \ N_1 \ N_2 \ ? \\
N_1 & \rightarrow & \text{chuck} \\
N_2 & \rightarrow & \text{wood} \\
N_3 & \rightarrow & \text{ould} \\
N_4 & \rightarrow & \text{a } N_2 N_1
\end{array}$$

$$\Downarrow$$

how much N_2 wN_3 N_4 N_1 if N_4 cN_3 N_1 N_2 | chuck | wood | ould | a N_2 N_1 |

Previous Approaches

1. Practical algorithms: Sequitur (and offline friends). 1996

   "Compression and Explanation Using Hierarchical Grammars". Nevill-Manning & Witten. The Computer Journal. 1997

2. Compression theoretical framework: Grammar Based Code. 2000

   "Grammar-based codes: a new class of universal lossless source codes". Kieffer & Yang. IEEE T on Information Theory. 2000

3. Approximation ratio to the size of a Smallest Grammar in the worst case. 2002

   "The Smallest Grammar Problem", Charikar et.al. IEEE T on Information Theory. 2005

Previous Approaches

1. Practical algorithms: Sequitur (and offline friends). 1996

   "Compression and Explanation Using Hierarchical Grammars". Nevill-Manning & Witten. The Computer Journal. 1997

2. Compression theoretical framework: Grammar Based Code. 2000

   "Grammar-based codes: a new class of universal lossless source codes". Kieffer & Yang. IEEE T on Information Theory. 2000

3. Approximation ratio to the size of a Smallest Grammar in the worst case. 2002

   "The Smallest Grammar Problem", Charikar et.al. IEEE T on Information Theory. 2005

Offline algorithms

- Maximal Length (ML): take longest repeat, replace all occurrences with new symbol, iterate.

Offline algorithms

- Maximal Length (ML): take longest repeat, replace all occurrences with new symbol, iterate.

$S \rightarrow$ how_much_wood_would_a_woodchuck_chuck_
if_a_woodchuck_could_chuck_wood?

Offline algorithms

- Maximal Length (ML): take longest repeat, replace all occurrences with new symbol, iterate.

$S \rightarrow$ how_much_wood_would_a_woodchuck_chuck_
if_a_woodchuck_could_chuck_wood?

Offline algorithms

- Maximal Length (ML): take longest repeat, replace all occurrences with new symbol, iterate.

$$S \rightarrow \text{how\_much\_wood\_would\_a\_woodchuck\_chuck\_if\_a\_woodchuck\_could\_chuck\_wood?}$$

$$\Downarrow$$

$$S \rightarrow \text{how\_much\_wood\_would}N_1\text{huck\_if}N_1\text{ould\_chuck\_wood?}$$

$$N_1 \rightarrow \text{\_a\_woodchuck\_c}$$

Offline algorithms

- Maximal Length (ML): take longest repeat, replace all occurrences with new symbol, iterate.

$S \rightarrow$ how_much_wood_would_a_woodchuck_chuck_
if_a_woodchuck_could_chuck_wood?

$\Downarrow$

$S \rightarrow$ how_much_wood_wouldN_1huck_ifN_1ould_chuck_wood?

$N_1 \rightarrow$ _a_woodchuck_c

Offline algorithms

- Maximal Length (ML): take longest repeat, replace all occurrences with new symbol, iterate.

S → how_much_wood_would_a_woodchuck_chuck_
if_a_woodchuck_could_chuck_wood?

⇓

S → how_much_wood_wouldN_1huck_ifN_1ould_chuck_wood?
N_1 → _a_woodchuck_c

⇓

S → how_much_wood_wouldN_1huck_if_N_1ould_N_2wood?
N_1 → _a_woodN_2c
N_2 → chuck_

Offline algorithms

- Maximal Length (ML): take longest repeat, replace all occurrences with new symbol, iterate.

  Bentley & McIlroy "Data compression using long common strings". DCC. 1999.

  Nakamura, et.al. "Linear-Time Text Compression by Longest-First Substitution". MDPI Algorithms. 1999

- Most Frequent (MF): take most frequent repeat, replace all occurrences with new symbol, iterate

  Larsson & Moffat. "Offline Dictionary-Based Compression". DCC. 1999

- Most Compressive (MC): take repeat that compress the best, replace with new symbol, iterate

  Apostolico & Lonardi. "Off-line compression by greedy textual substitution" Proceedings of IEEE. 2000

A General Framework: IRR

IRR (Iterative Repeat Replacement) framework
Input: a sequence s, a score function f

1. Initialize Grammar by $S \rightarrow s$
2. take repeat ω that maximizes f over G
3. **if** replacing ω would yield a bigger grammar than G **then**
   3.1 **return** G

   else
   3.1 replace all (non-overlapping) occurrences of ω in G by new symbol N
   3.2 add rule $N \rightarrow \omega$ to G
   3.3 goto 2

Complexity: $\mathcal{O}(n^3)$

Results on Canterbury Corpus

sequence	Sequitur	IRR-ML	IRR-MF	IRR-MC
alice29.txt	19.9%	37.1%	8.9%	41000
asyoulik.txt	17.7%	37.8%	8.0%	37474
cp.html	22.2%	21.6%	10.4%	8048
fields.c	20.3%	18.6%	16.1%	3416
grammar.lsp	20.2%	20.7%	15.1%	1473
kennedy.xls	4.6%	7.7%	0.3%	166924
lcet10.txt	24.5%	45.0%	8.0%	90099
plrabn12.txt	14.9%	45.2%	5.8%	124198
ptt5	23.4%	26.1%	6.4%	45135
sum	25.6%	15.6%	11.9%	12207
xargs.1	16.1%	16.2%	11.8%	2006
average	*19.0%*	*26.5%*	*9.3%*	

Extends and confirms results of Nevill-Manning & Witten "On-Line and Off-Line Heuristics for Inferring Hierarchies of Repetitions in Sequences". Proc. of the IEEE. vol 80 no 11. November 2000

A General Framework: IRR

IRR (Iterative Repeat Replacement) framework
Input: a sequence s, a score function f

1. Initialize Grammar by $S \rightarrow s$
2. take repeat ω that maximizes f over G
3. **if** replacing ω would yield a bigger grammar than G **then**
   3.1 **return** G

   else

   3.1 replace all (non-overlapping) occurrences of ω in G by new symbol N
   3.2 add rule $N \rightarrow \omega$ to G
   3.3 goto 2

Complexity: $\mathcal{O}(n^3)$

Split the Problem

Smallest Grammar Problem → Choice of Constituents

Smallest Grammar Problem → Choice of Occurrences

A General Framework: IRR

IRR (Iterative Repeat Replacement) framework
Input: a sequence s, a score function f

1. Initialize Grammar by $S \rightarrow s$
2. take repeat ω that maximizes f over G
3. **if** replacing ω would yield a bigger grammar than G **then**
   3.1 **return** G

   else

   3.1 replace all (non-overlapping) occurrences of ω in G by new symbol N
   3.2 add rule $N \rightarrow \omega$ to G
   3.3 goto 2

Complexity: $\mathcal{O}(n^3)$

A General Framework: IRRCOO

IRRCOO (Iterative Repeat Replacement with Choice of Occurrence Optimization) framework

Input: a sequence s, a score function f

1. Initialize Grammar by $S \rightarrow s$
2. take repeat ω that maximizes f over G
3. **if** replacing ω would yield a bigger grammar than G **then**
   3.1 **return** G

   else

   3.1 $G \leftarrow mgp(cons(G) \cup cons(\omega))$

   3.2 goto 2

Choice of Occurrences

Minimal Grammar Parsing (MGP) Problem

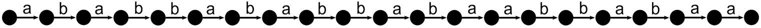
Given sequences $\Omega = \{s = w_0, w_1, \ldots, w_m\}$, find a context-free grammar of minimal size that has non-terminals $\{S = N_0, N_1, \ldots N_m\}$ such that $\text{cons}(N_i) = w_i$.

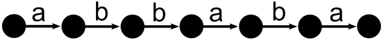
Choice of Occurrences: an Example

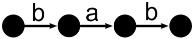
Given sequences $\Omega = \{ababbababbabaabbabaa, abbaba, bab\}$

Choice of Occurrences: an Example

Given sequences $\Omega = \{ababbababbabaabbabaa, abbaba, bab\}$

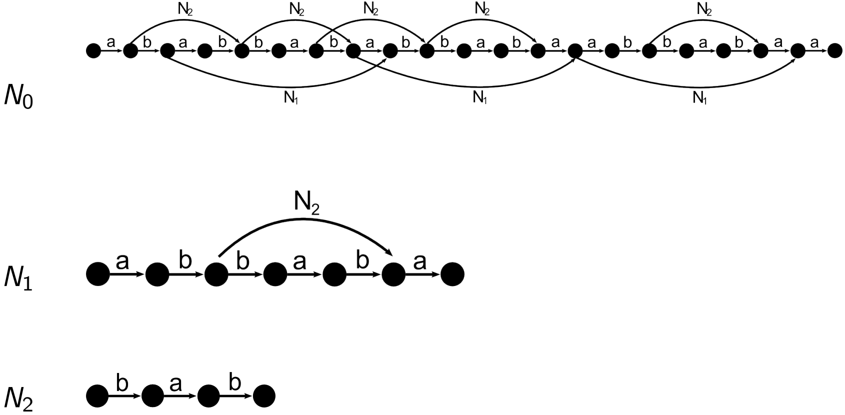
N_0 

N_1 

N_2 

Choice of Occurrences: an Example

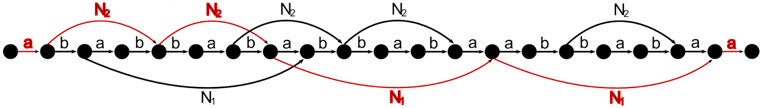
Given sequences $\Omega = \{ababbababbabaabbabaa, abbaba, bab\}$



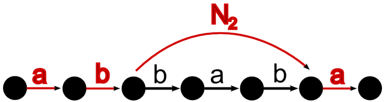
Choice of Occurrences: an Example

Given sequences $\Omega = \{ababbababbabaabbabaa, abbaba, bab\}$

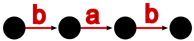
N_0



N_1



N_2



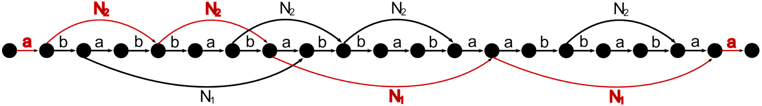
A minimal grammar for Ω is

$$
\begin{aligned}
N_0 &\rightarrow aN_2N_2N_1N_1a \\
N_1 &\rightarrow abN_2a \\
N_2 &\rightarrow bab
\end{aligned}
$$

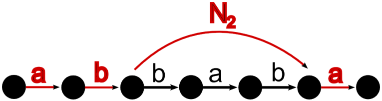
Choice of Occurrences: an Example

Given sequences $\Omega = \{ababbababbabaabbabaa, abbaba, bab\}$

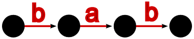
N_0



N_1



N_2



mgp can be computed in $\mathcal{O}(n^3)$

Split the Problem

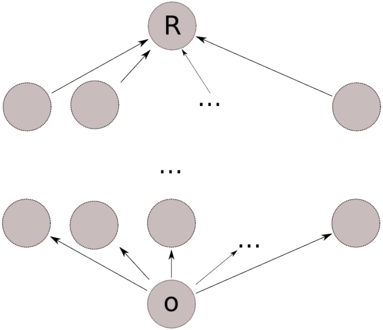
Smallest Grammar Problem

Choice of Constituents

Choice of Occurrences

A Search Space for the SGP

Given s, take the lattice $\langle \mathcal{R}(s), \subseteq \rangle$ and associate a score to each node η: the size of the grammar $mgp(\eta \cup \{s\})$. A smallest grammar will have associated a node with minimal score.



A Search Space for the SGP

Lattice is a good search space

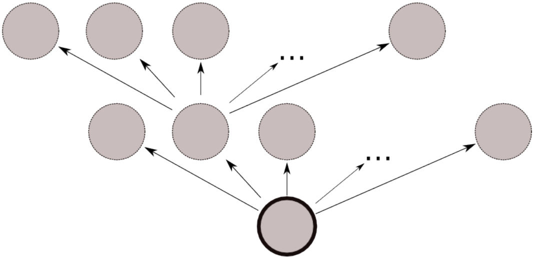
For every sequence s, there is a node η in $\langle \mathcal{R}(s), \subseteq \rangle$ such that $mgp(\eta \cup \{s\})$ is a smallest grammar.

Not the case for IRR search space

But, there exists a sequence s such that for any score function f, $IRR(s, f)$ does not return a smallest grammar ▸ Proof

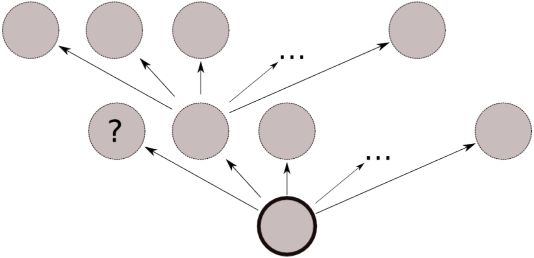
The ZZ Algorithm

bottom-up phase: given node η, compute scores of nodes $\eta \cup \{w_i\}$ and take node with smallest score.



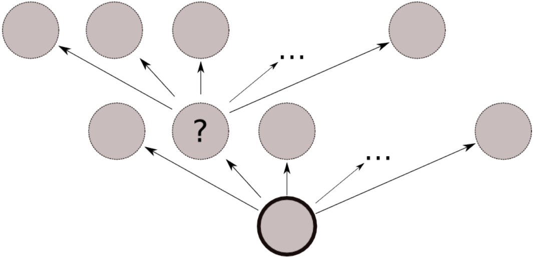
The ZZ Algorithm

bottom-up phase: given node η, compute scores of nodes $\eta \cup \{w_i\}$ and take node with smallest score.



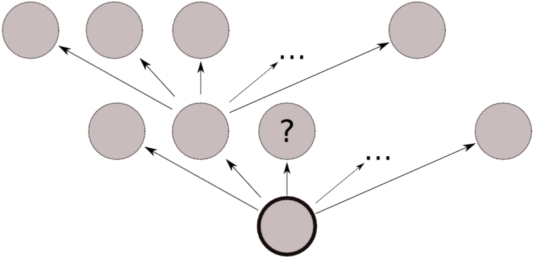
The ZZ Algorithm

bottom-up phase: given node η, compute scores of nodes $\eta \cup \{w_i\}$ and take node with smallest score.



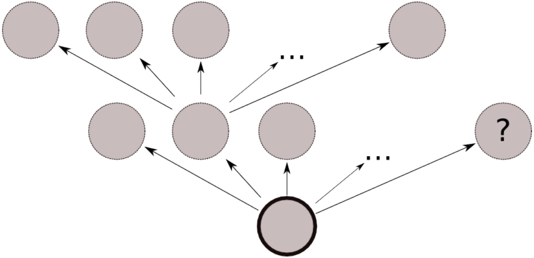
The ZZ Algorithm

bottom-up phase: given node η, compute scores of nodes $\eta \cup \{w_i\}$ and take node with smallest score.



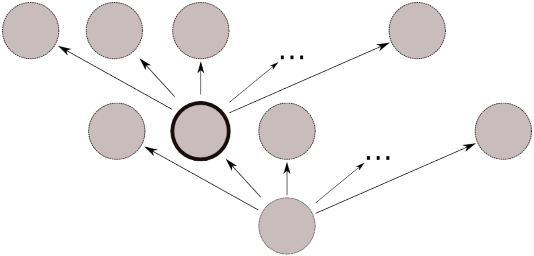
The ZZ Algorithm

bottom-up phase: given node η, compute scores of nodes $\eta \cup \{w_i\}$ and take node with smallest score.



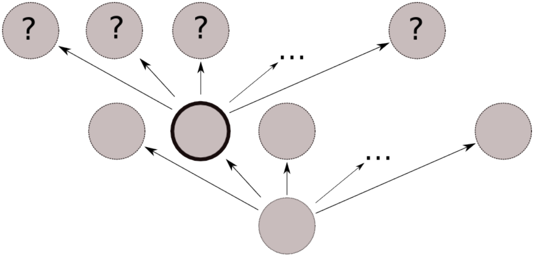
The ZZ Algorithm

bottom-up phase: given node η, compute scores of nodes $\eta \cup \{w_i\}$ and take node with smallest score.



The ZZ Algorithm

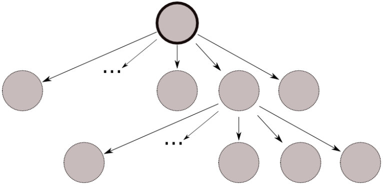
bottom-up phase: given node η, compute scores of nodes $\eta \cup \{w_i\}$ and take node with smallest score.



The ZZ Algorithm

bottom-up phase: given node η, compute scores of nodes $\eta \cup \{w_i\}$ and take node with smallest score.

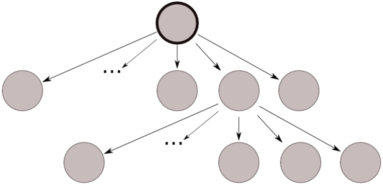
top-down phase: given node η, compute scores of nodes $\eta \setminus \{w_i\}$ and take node with smallest score.



The ZZ Algorithm

bottom-up phase: given node η, compute scores of nodes $\eta \cup \{w_i\}$ and take node with smallest score.

top-down phase: given node η, compute scores of nodes $\eta \setminus \{w_i\}$ and take node with smallest score.



ZZ: succession of both phases. Is in $\mathcal{O}(n^7)$

Results on Canterbury Corpus

sequence	IRRCOO-MC	ZZ	IRR-MC
alice29.txt	-4.3%	-8.0%	41000
asyoulik.txt	-2.9%	-6.6%	37474
cp.html	-1.3%	-3.5%	8048
fields.c	-1.3%	-3.1%	3416
grammar.lsp	-0.1%	-0.5%	1473
kennedy.xls	-0.1%	-0.1%	166924
lcet10.txt	-1.7%	–	90099
plrabn12.txt	-5.5%	–	124198
ptt5	-2.6%	–	45135
sum	-0.8%	-1.5%	12207
xargs.1	-0.8%	-1.7%	2006
average	*-2.0%*	*-3.1%*	

New Results

Classi-fication	sequence name	length	IRRMGP*	size im-provement
Virus	P. lambda	48 Knt	13061	-4.25%
Bacterium	E. coli	4.6 Mnt	741435	-8.82%
Protist	T. pseudonana chrI	3 Mnt	509203	-8.15%
Fungus	S. cerevisiae	12.1 Mnt	1742489	-9.68%
Alga	O. tauri	12.5 Mnt	1801936	-8.78%

Back to Structure

How similar are the structures returned by the different algorithms?

Back to Structure

How similar are the structures returned by the different algorithms?
Standard measure to compare parse trees:

- Unlabeled Precision and Recall (F-measure)
- Unlabeled Non Crossing Precision and Recall (F-measure)

Dan Klein. "The Unsupervised Learning of Natural Language Structure". Phd Thesis. U Stanford. 2005

Similarity of Structure

sequence	algorithm vs IRR-MC	size gain	U_F	UNC_F
fields.c	ZZ	3.1 %	77.8	85.3
	IRRCOO-MC	1.3 %	84.1	88.7
cp.html	ZZ	3.5 %	66.3	75.0
	IRRCOO-MC	1.3 %	81.4	84.8
alice.txt	ZZ	8.0 %	36.6	38.6
	IRRCOO-MC	4.3 %	63.9	66.0
asyoulike.txt	ZZ	6.6 %	34.6	35.8
	IRRCOO-MC	2.9 %	55.1	56.9

Conclusions and Perspectives

- Split SGP into two complementary problems: choice of constituents and choice of occurrences
- Definition of a search space that contains a solution....
- ... and to define algorithms which find smaller grammars than state-of-the-art.

Conclusions and Perspectives

- ⋆ Split SGP into two complementary problems: choice of constituents and choice of occurrences
- ⋆ Definition of a search space that contains a solution....
- ⋆ ... and to define algorithms which find smaller grammars than state-of-the-art.
- Promising results on DNA sequences (whole genomes)

Conclusions and Perspectives

- ⋆ Split SGP into two complementary problems: choice of constituents and choice of occurrences
- ⋆ Definition of a search space that contains a solution....
- ⋆ ... and to define algorithms which find smaller grammars than state-of-the-art.
- Promising results on DNA sequences (whole genomes)
- Focus on the structure. Meaning of (dis)similarity.

The End

S → thDkAforBr_attenC._DoAhave_Dy_quesCs?
A → B_
B → _you
C → tion
D → an

Parse Tree Similarity Measures

$$UNC_P(P_1, P_2) = \frac{|\{b \in \text{brackets}(P_1) : b \text{ does not cross brackets}(P_2)|}{|\text{brackets}(P_1)|}$$

$$UNC_R(P_1, P_2) = \frac{|\{b \in \text{brackets}(P_2) : b \text{ does not cross brackets}(P_1)|}{|\text{brackets}(P_2)|}$$

$$UNC_F(P_1, P_2) = \frac{2}{UNC_P(P_1, P_2)^{-1} + UNC_R(P_1, P_2)^{-1}}$$

Parse Tree Similarity Measures

$$U_P(P_1, P_2) = \frac{|\text{brackets}(P_1) \cap \text{brackets}(P_2)|}{|\text{brackets}(P_1)|}$$

$$U_R(P_1, P_2) = \frac{|\text{brackets}(P_1) \cap \text{brackets}(P_2)|}{|\text{brackets}(P_2)|}$$

$$U_F(P_1, P_2) = \frac{2}{U_P(P_1, P_2)^{-1} + U_R(P_1, P_2)^{-1}}$$

Problems of IRR-like algorithms

Example

$xaxbxcx|_1 xbxcxax|_2 xcxaxbx|_3 xaxcxbx|_4 xbxaxcx|_5 xcxbxax|_6 xax|_7 xbx|_8 xcx$

Problems of IRR-like algorithms

Example

$xaxbxcx|_1 xbxcxax|_2 xcxaxbx|_3 xaxcxbx|_4 xbxaxcx|_5 xcxbxax|_6 xax|_7 xbx|_8 xcx$

A smallest grammar is:

$$\begin{aligned} S &\rightarrow AbC|_1 BcA|_2 CaB|_3 AcB|_4 BaC|_5 CbA|_6 A|_7 B|_8 C \\ A &\rightarrow xax \\ B &\rightarrow xbx \\ C &\rightarrow xcx \end{aligned}$$

Problems of IRR-like algorithms

Example

$xaxbxcx|_1 xbxcxax|_2 xcxaxbx|_3 xaxcxbx|_4 xbxaxcx|_5 xcxbxax|_6 xax|_7 xbx|_8 xcx$

But what IRR can do is like:

$$
\begin{array}{lcl}
S & \rightarrow & Abxcx|_1 xbxcA|_2 xcAbx|_3 Acxbx|_4 xbAcx|_5 xcxbA|_6 A|_7 xbx|_8 xcx \\
A & \rightarrow & xax \\
 & \Downarrow & \\
S & \rightarrow & Abxcx|_1 BcA|_2 xcAbx|_3 AcB|_4 xbAcx|_5 xcxbA|_6 A|_7 B|_8 xcx \\
A & \rightarrow & xax \\
B & \rightarrow & xbx \\
 & \Downarrow & \\
S & \rightarrow & AbC|_1 BcA|_2 xcAbx|_3 AcB|_4 xbAcx|_5 CbA|_6 A|_7 B|_8 C \\
A & \rightarrow & xax \\
B & \rightarrow & xbx \\
C & \rightarrow & xcx
\end{array}
$$