Coupled Transformations of SPPFs

Dr. Vadim Zaytsev aka @grammarware

GCM @ STAF 2015

SLEBoK — BibSLEIGH —

bibtex.github.io/person/Andy_Schuerr.html

@BIBSLEIGH

corpus
tags
bundles
people
edit!

Person: Andy Schürr

DBLP: Sch=uuml=rr:Andy

Facilitated 5 volumes:

Ed SciCo Ed SciCo SciCo

Contributed to:

2014 2014 2014 2014 2013 2012 2012 2015 2012 2012 2012 2011 2011 2009 2008 2008 2008 2007

2008 2006 2006 2002 2000 1998 1997 1997 1997 1995 1994 1990 1987

Wrote 40 papers:

ECMFA-2014-AnjorinRDS T T
Efficient Model Synchronization with View Triple Graph Grammars (AA, SR, FD, AS), pp. 1–17.
FASE-2014-AnjorinSLS T
Modularizing Triple Graph Grammars Using Rule Refinement (AA, KS, ML, AS), pp. 340–354.
ICGT-2014-AnjorinLST T T T T
A Static Analysis of Non-confluent Triple Graph Grammars for Efficient Model Transformation (AA, EL, AS, GT), pp. 130–145.
ICMT-2014-LeblebiciAS
Developing eMoflon with eMoflon (EL, AA, AS), pp. 138–145.
ICMT-2014-WieberAS T T T
On the Usage of TGGs for Automated Model Transformation Testing (MW, AA, AS), pp. 1–16.
ICMT-2013-WieberS T T
Systematic Testing of Graph Transformations: A Practical Approach Based on Graph Patterns (MW, AS), pp. 205–220.
ECMFA-2012-LauderAVS T T T
Bidirectional Model Transformation with Precedence Triple Graph Grammars (ML, AA, GV, AS), pp. 287–302.
ECMFA-2012-VarroAS T T
Unification of Compiled and Interpreter-Based Pattern Matching Techniques (GV, AA, AS), pp. 368–383.
ICGT-2012-AnjorinST T
Construction of Integrity Preserving Triple Graph Grammars (AA, AS, GT), pp. 356–370.
ICGT-2012-LauderAVS T T
Efficient Model Synchronization with Precedence Triple Graph Grammars (ML, AA, GV, AS), pp. 401–415.

Travelled to:

1 ⁄ Australia
1 ⁄ Brazil
1 ⁄ Croatia
1 ⁄ Czech Republic
1 ⁄ Denmark
3 ⁄ France
5 ⁄ Germany
1 ⁄ Hungary
1 ⁄ New Zealand
3 ⁄ Spain
2 ⁄ Switzerland
2 ⁄ USA
4 ⁄ United Kingdom

Collaborated with:

Ø A. Anjorin A.J. Winter G. Varró M. Wieber A. Königs F. Klar M. Nagl C. Amelunxen T. Rötschke F. Deckwerth M. Lauder G. Tántzer E. Leblebici J. Weisemöller M. Lochau K. Saller G. Engels S. Rose A. Winter A. Zündorf A. Borkowski M. Münch J. Szuba R.C. Holt J. Jakob H. Cichos S. Oster D. Wagelaar D.S. Kolovos M. Wimmer W. Retschitzegger A. Kusel W. Schwinger G. Kappel J. Schönböck R.F. Paige J.N. Foster Z. Hu K. Czarnecki J.F. Terwilliger R. Lämmel

SLEBoK — BibSLEIGH

bibtex.github.io/person/Vadim_Zaytsev.html

@BIBSLEIGH

corpus
tags
bundles
people
edit!

Person: Vadim Zaytsev

DBLP: Zaytsev:Vadim

Facilitated 7 volumes:

PubCh SciCo PrCo PrCo PubCh PubCh WebCh

Contributed to:

2014 2014 2013 2013 2012 2012 2011 2010 2009 2009 2011 2009

Wrote 13 papers:

CSMR-WCRE-2014-BaggeZ
International workshop on open and original problems in software language engineering (AHB, VZ), p. 478.

CSMR-WCRE-2014-Zaytsev T T
Formal foundations for semi-parsing (VZ), pp. 313–317.

MODELS-2014-ZaytsevB T T
Parsing in a Broad Sense (VZ, AHB), pp. 50–67.

SLE-2013-Zaytsev
Micropatterns in Grammars (VZ), pp. 117–136.

WCRE-2013-BaggeZ
Workshop on open and original problems in software language engineering (AHB, VZ), pp. 493–494.

LDTA-2012-Zaytsev T
Notation-parametric grammar recovery (VZ), p. 9.

SAC-2012-Zaytsev T
BNF was here: what have we done about the unnecessary diversity of notation for syntactic definitions (VZ), pp. 1910–1915.

SLE-2011-FischerLZ T T T
Comparison of Context-Free Grammars Based on Parsing Generated Test Data (BF, RL, VZ), pp. 324–343.

SLE-2010-ZaytsevL T
A Unified Format for Language Documents (VZ, RL), pp. 206–225.

GTTSE-2009-Zaytsev T
Language Convergence Infrastructure (VZ), pp. 481–497.

IFM-2009-LammelZ
An Introduction to Grammar Convergence (RL, VZ), pp. 246–260.

SCAM-2009-J-LammelZ11 T T
Recovering grammar relationships for the Java Language Specification (RL, VZ), pp. 333–378.

Travelled to:

1 ⁄ Belgium
1 ⁄ Canada
1 ⁄ Estonia
2 ⁄ Germany
1 ⁄ Italy
2 ⁄ Portugal
1 ⁄ Spain
1 ⁄ The Netherlands
1 ⁄ USA

Collaborated with:

Ø R. Lämmel
A.H. Bagge
B. Fischer

Vadim Zaytsev (@gramma

Vadim

Twitter, Inc. [US] https://twitter.com/grammarware

Home 16 Notifications Messages

Search Twitter

Tweet

TWEETS 13.3K | FOLLOWING 3,696 | FOLLOWERS 3,465 | FAVORITES 214 | LISTS 16

Edit profile

Vadim Zaytsev

@grammarware

language engineering freak, university maniac, programmer, hacker, automation enthusiast, wiki addict, grammar nazi, blues fan

Yurrup

grammarware.net

Joined April 2009

520 Photos and videos

Tweets | Tweets & replies | Photos & videos

Vadim Zaytsev @grammarware · 4m

With hardware in the loop, you can run less tests than with software in the loop and much less than with models (Lionel Briand #staf15)

Vadim Zaytsev @grammarware · 7m

Testing software controllers in the automative domain is the first case study presented by Lionel Briand at #staf15

1

Vadim Zaytsev @grammarware · 8m

Realistic solutions vs universal solutions: the struggle is real. (Lionel Briand #staf15)

Vadim Zaytsev @grammarware · 9m

Lionel Briand uses “verification” to mean defect detection and removal, before or after deployment; it’s verification in a broad sense!

Who to follow · Refresh · View all

GPCE @GPCECONF
Followed by Pjotr Kourzano…
Follow

Joshua Kerievsky @Joshua…
Follow

Carlos Noguera @CFNoguera
Follow

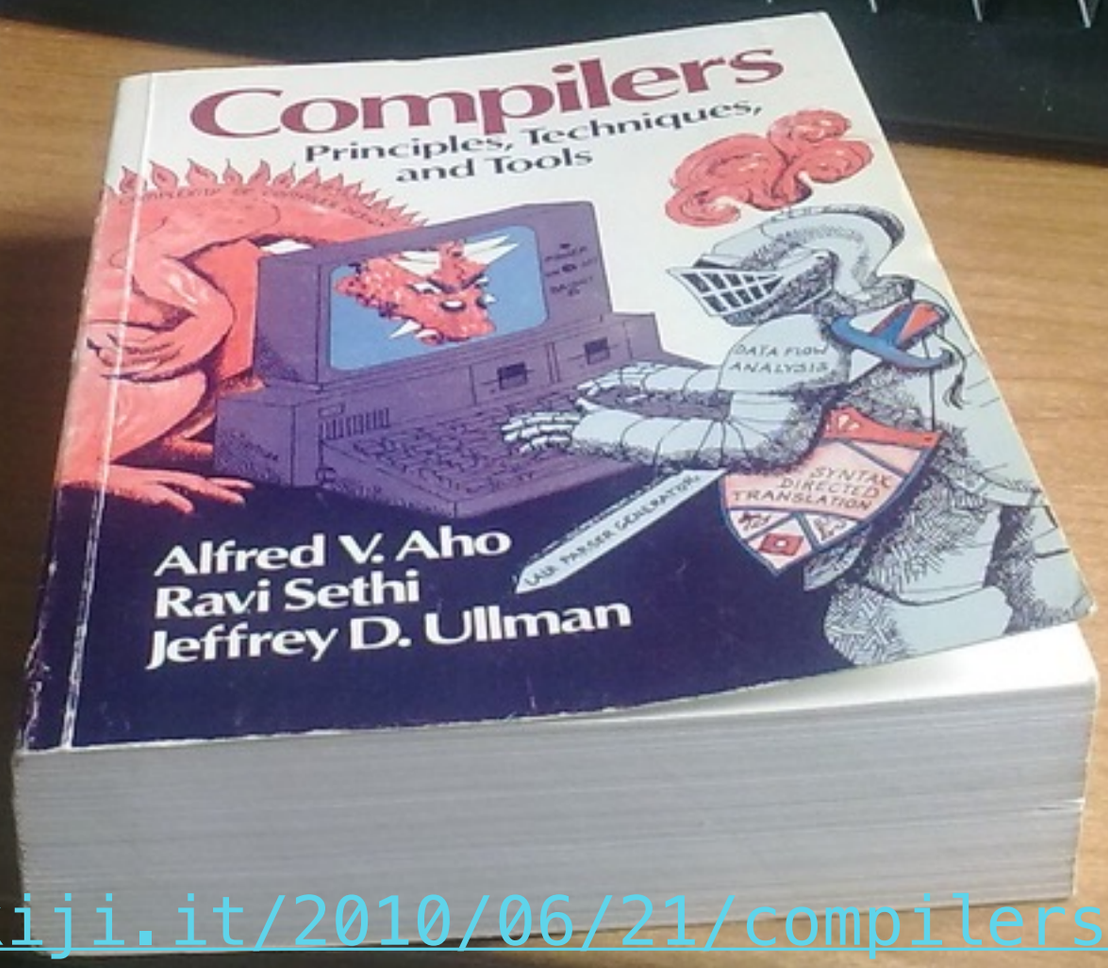
Find friends

Trends · Change

#GrowingUpShy
Carter
#LiamAppreciationDay
#WWEBattleground
FB PLS
Ty Lawson
#Feyenoord
BET Uncut
#rtlvis
Ashley Madison

Grammars

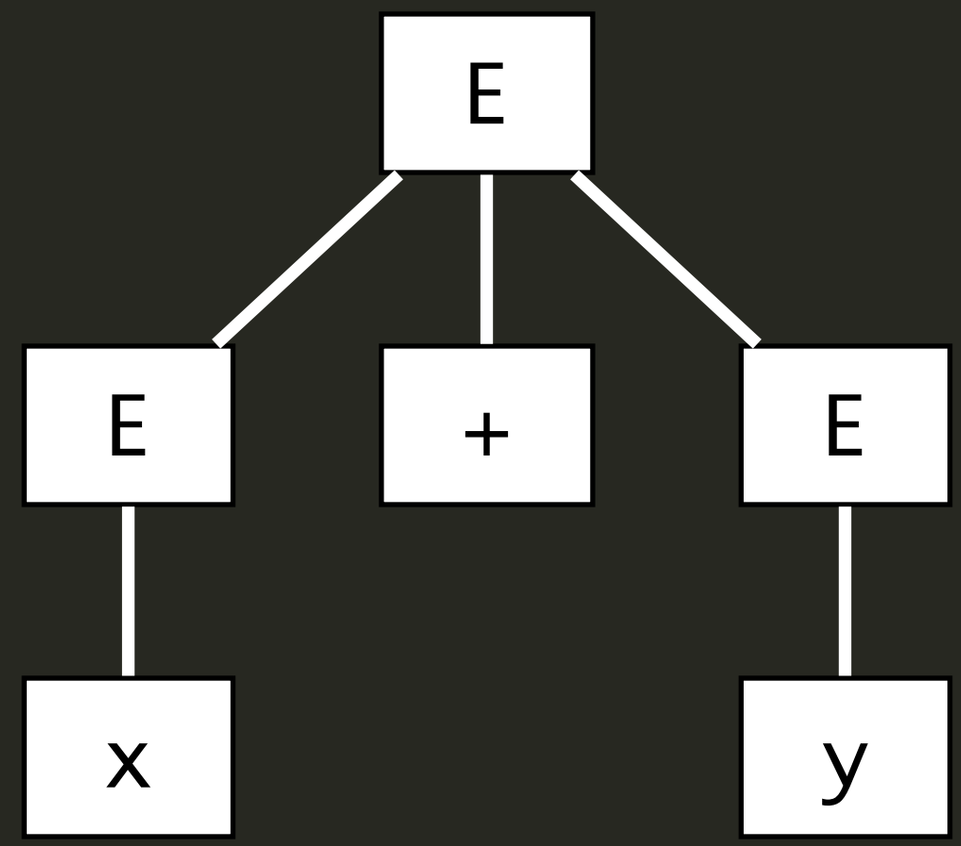
- ✓ Finite language definitions
- ✓ Text biased
- ✓ Tree biased
- ✓ Can be generalised



http://rikiji.it/2010/06/21/compilers.html

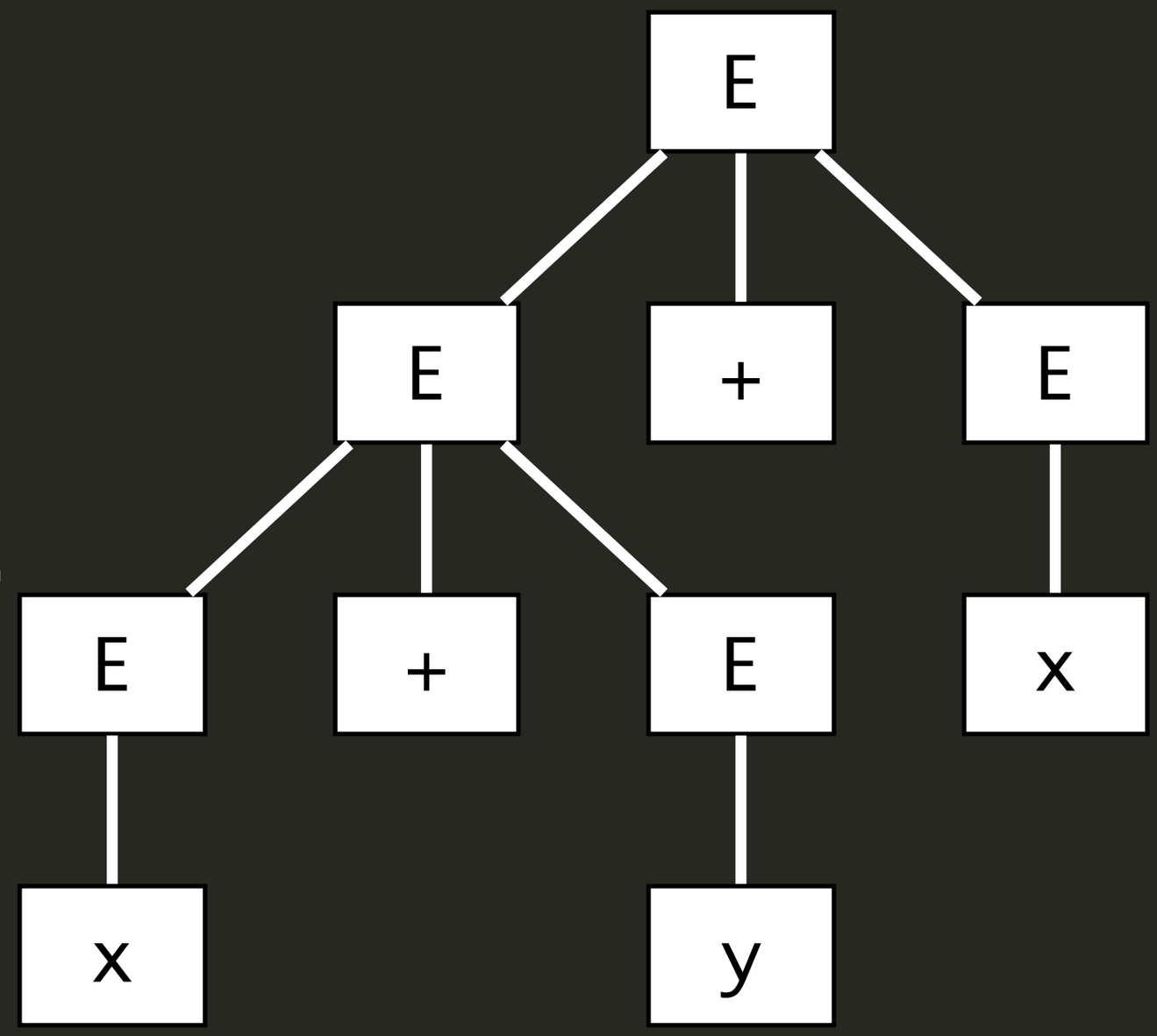
Grammars ⇒ Trees

```
E ::= E "+" E;

E ::= "x" | "y";
```

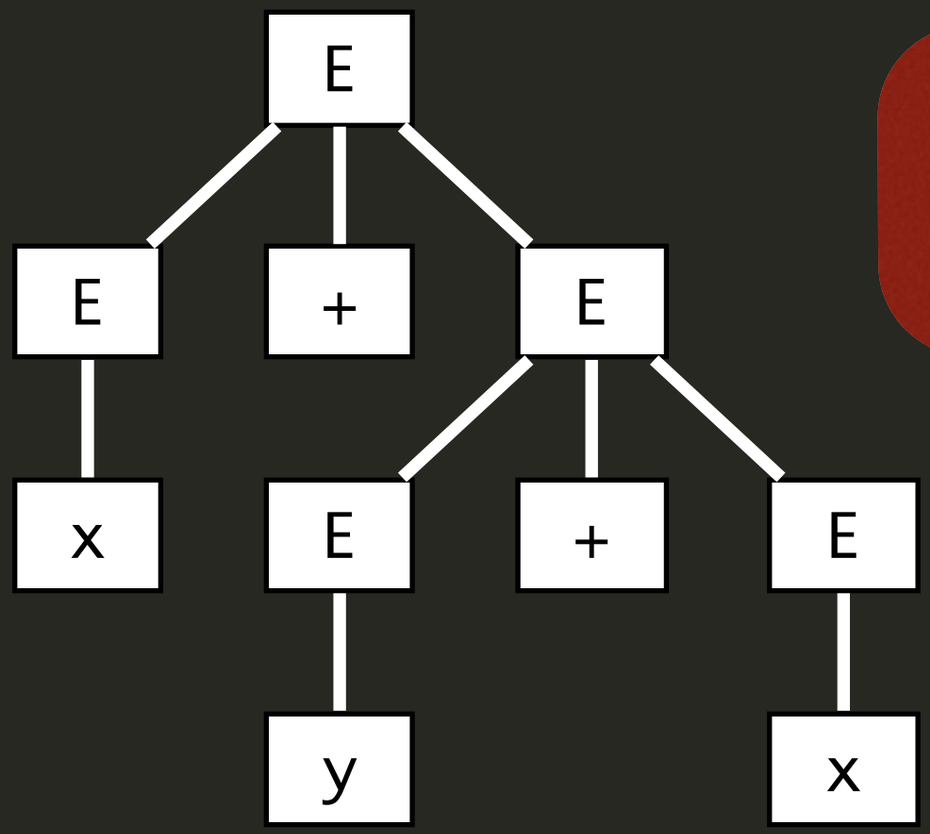


Grammars ⇒ Trees

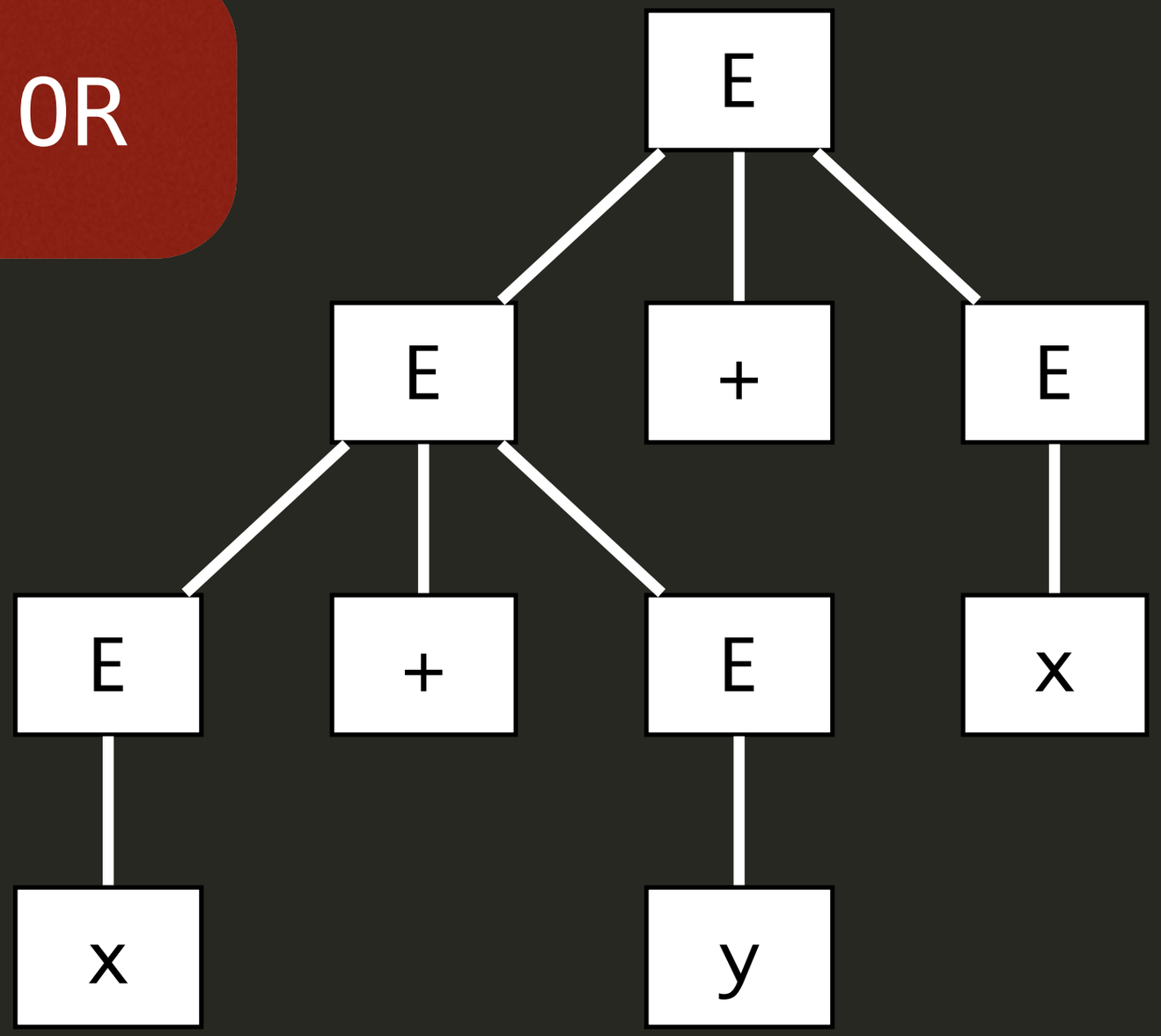
```
E ::= E "+" E;

E ::= "x" | "y";
```



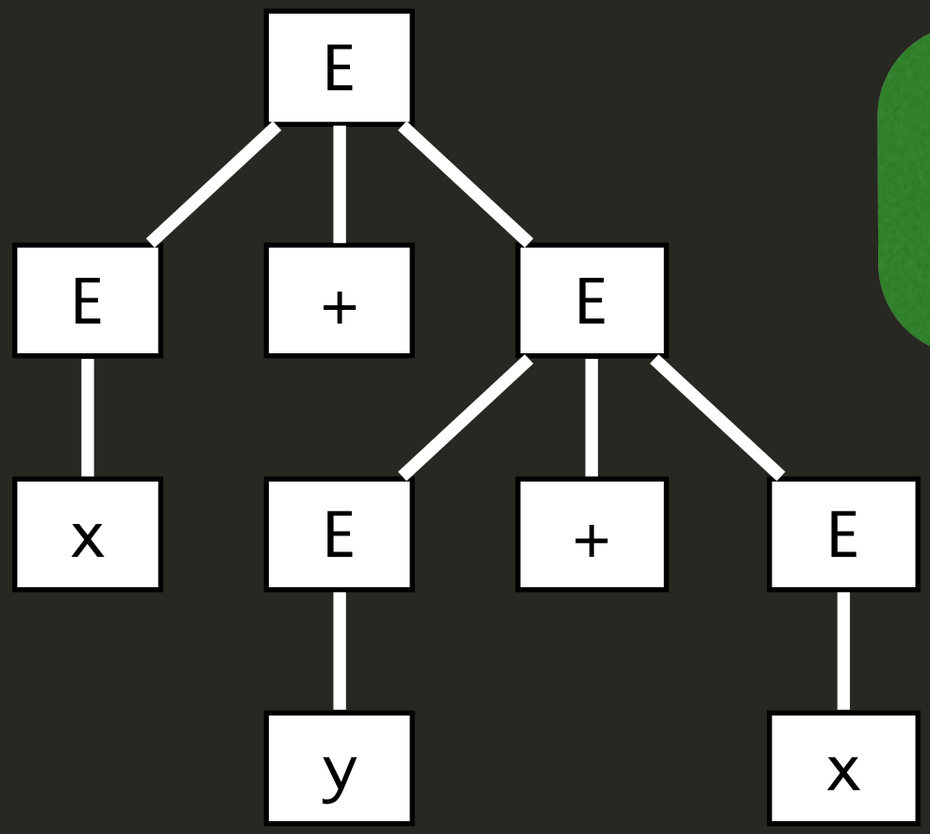
Grammars ⇒ Trees



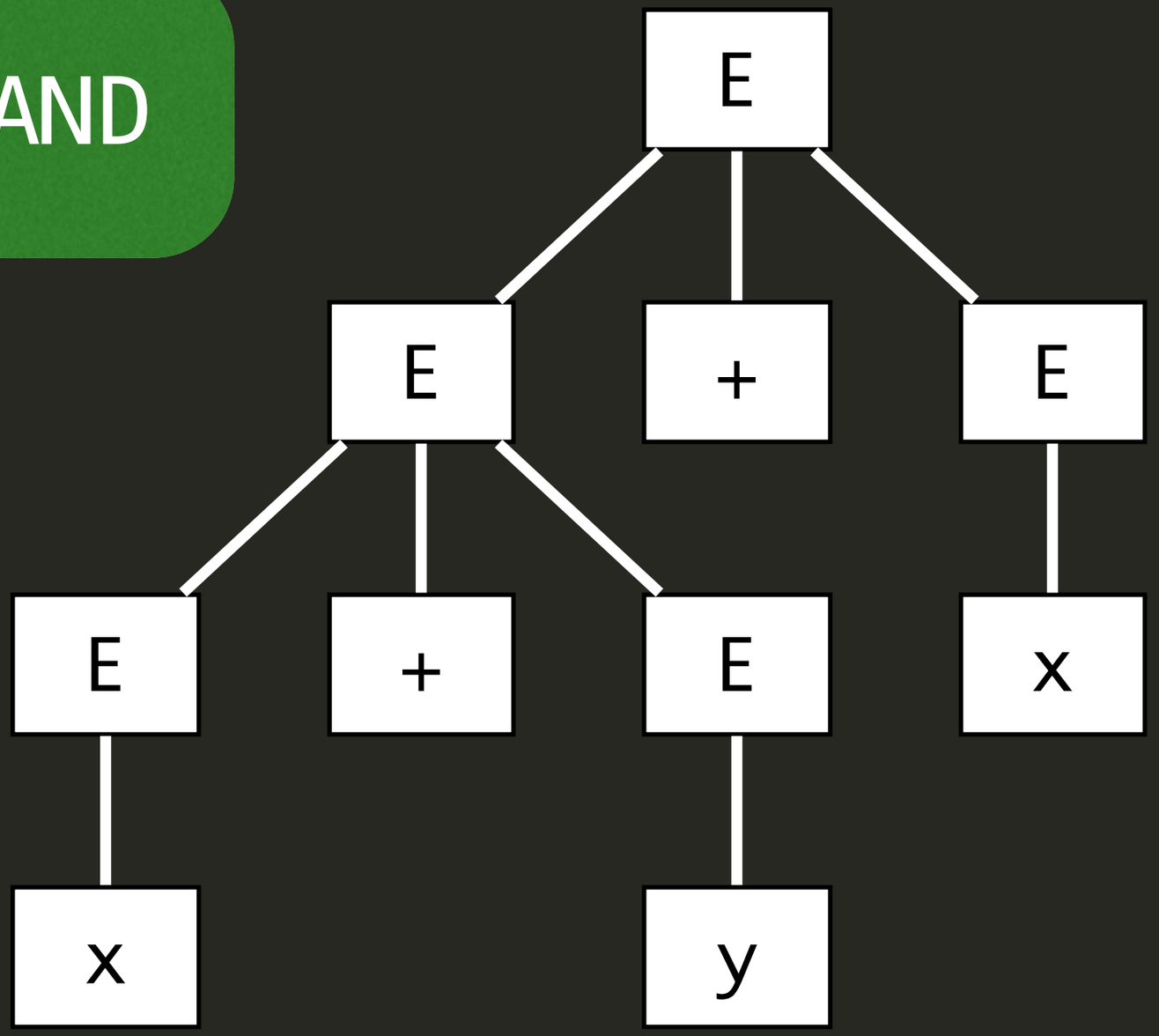
OR



Grammars ⇒ Trees



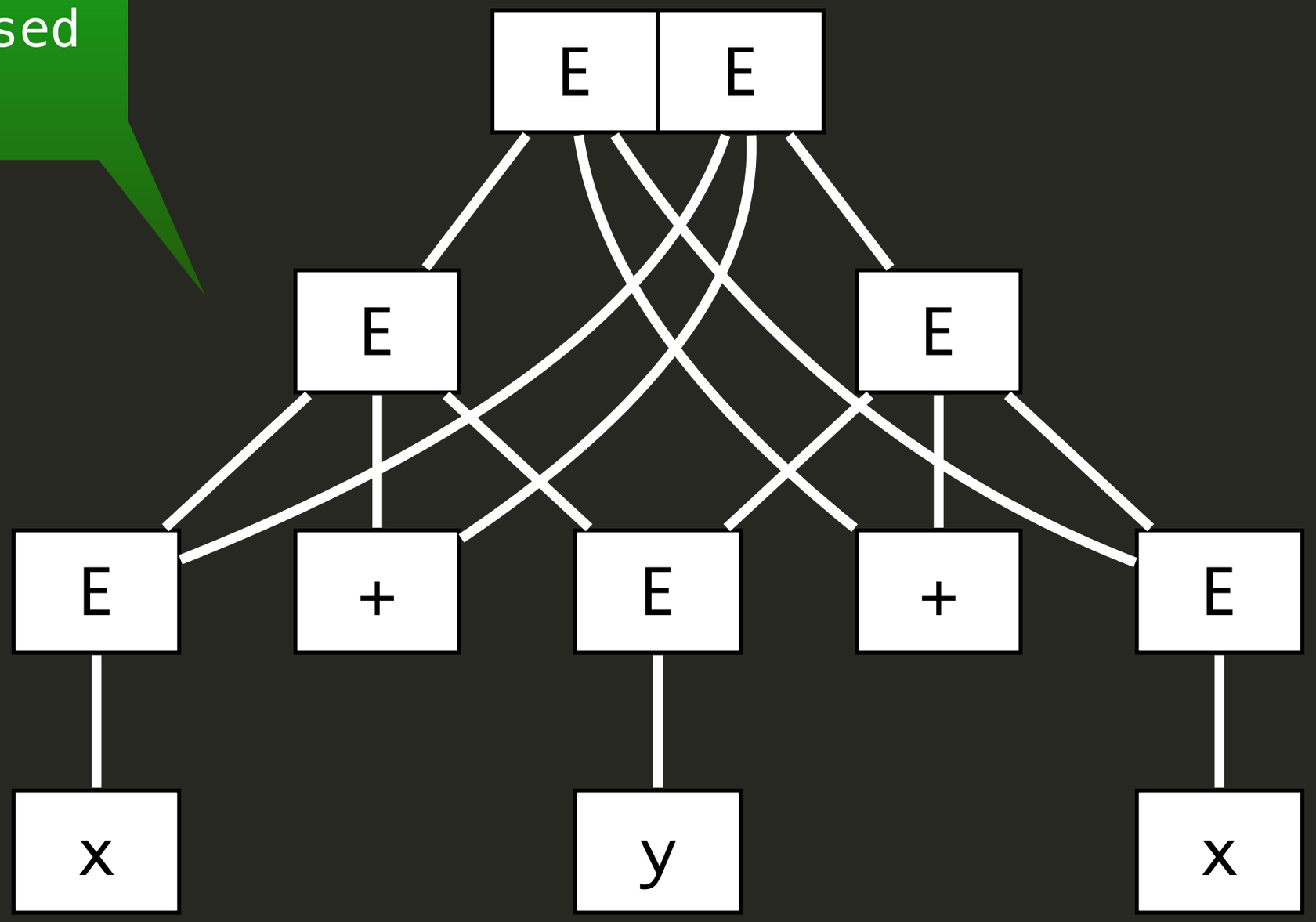
AND



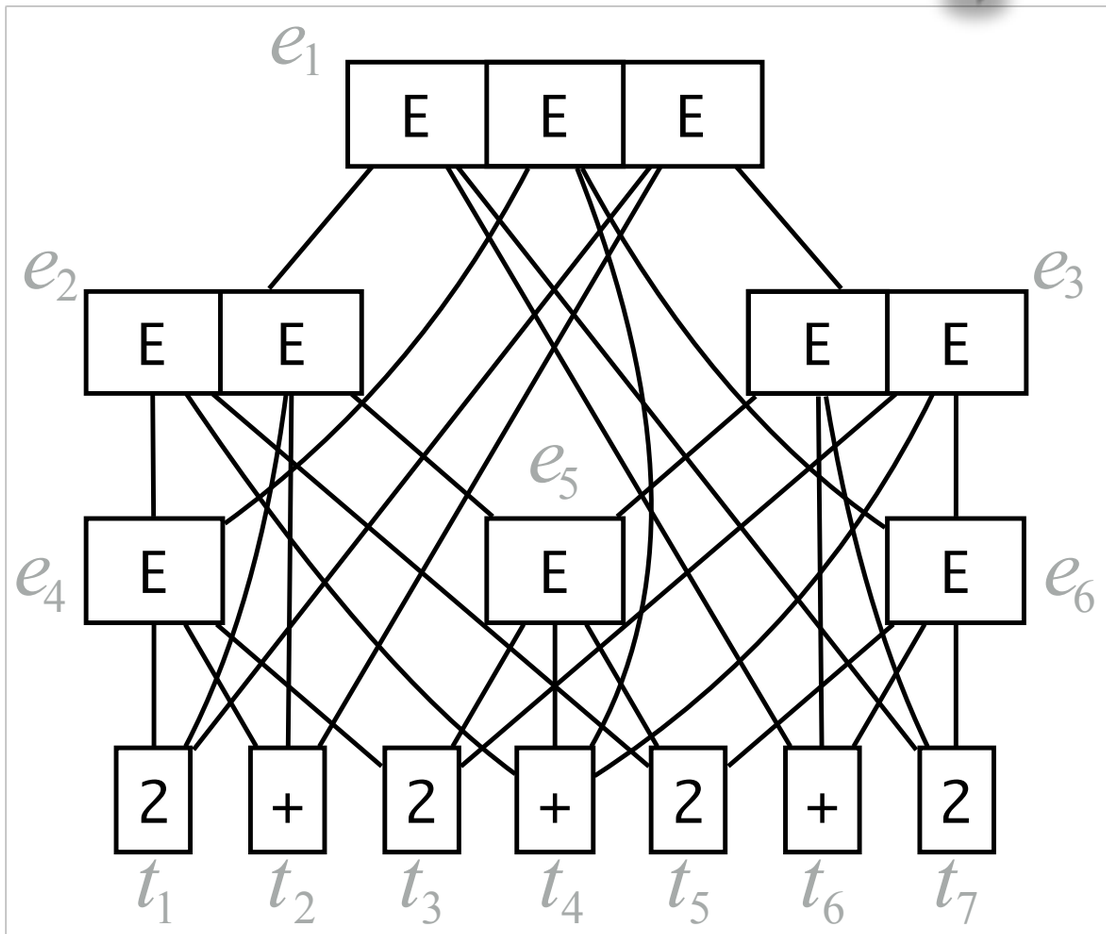
generalised parsing

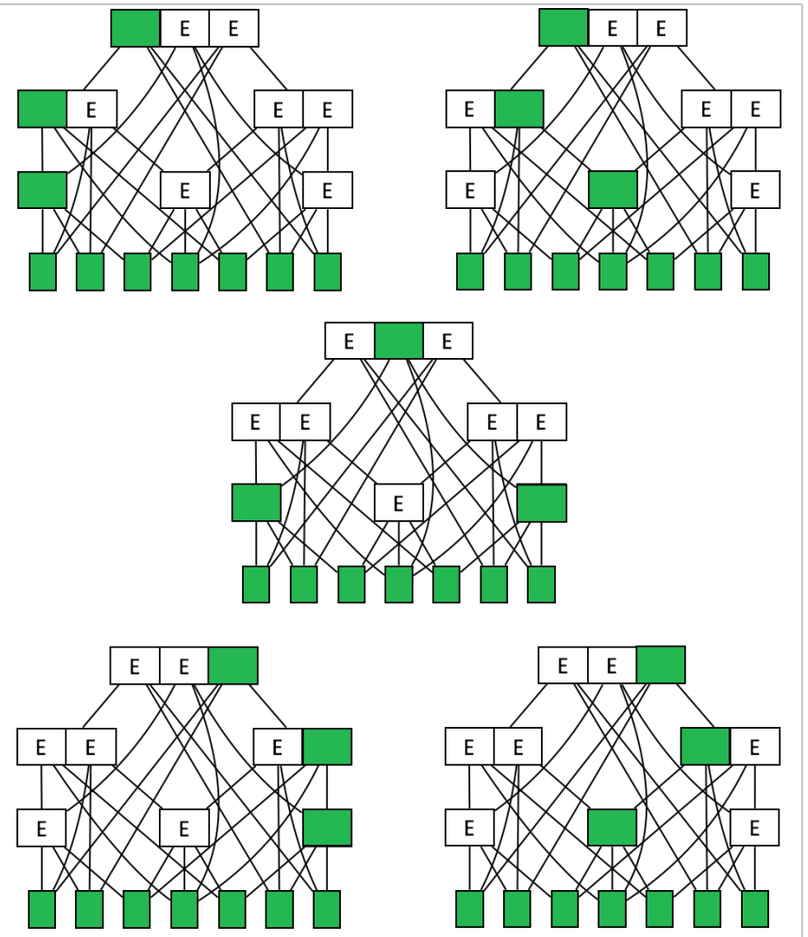
Grammars ⇒ Graphs

generalised parse



SPPF





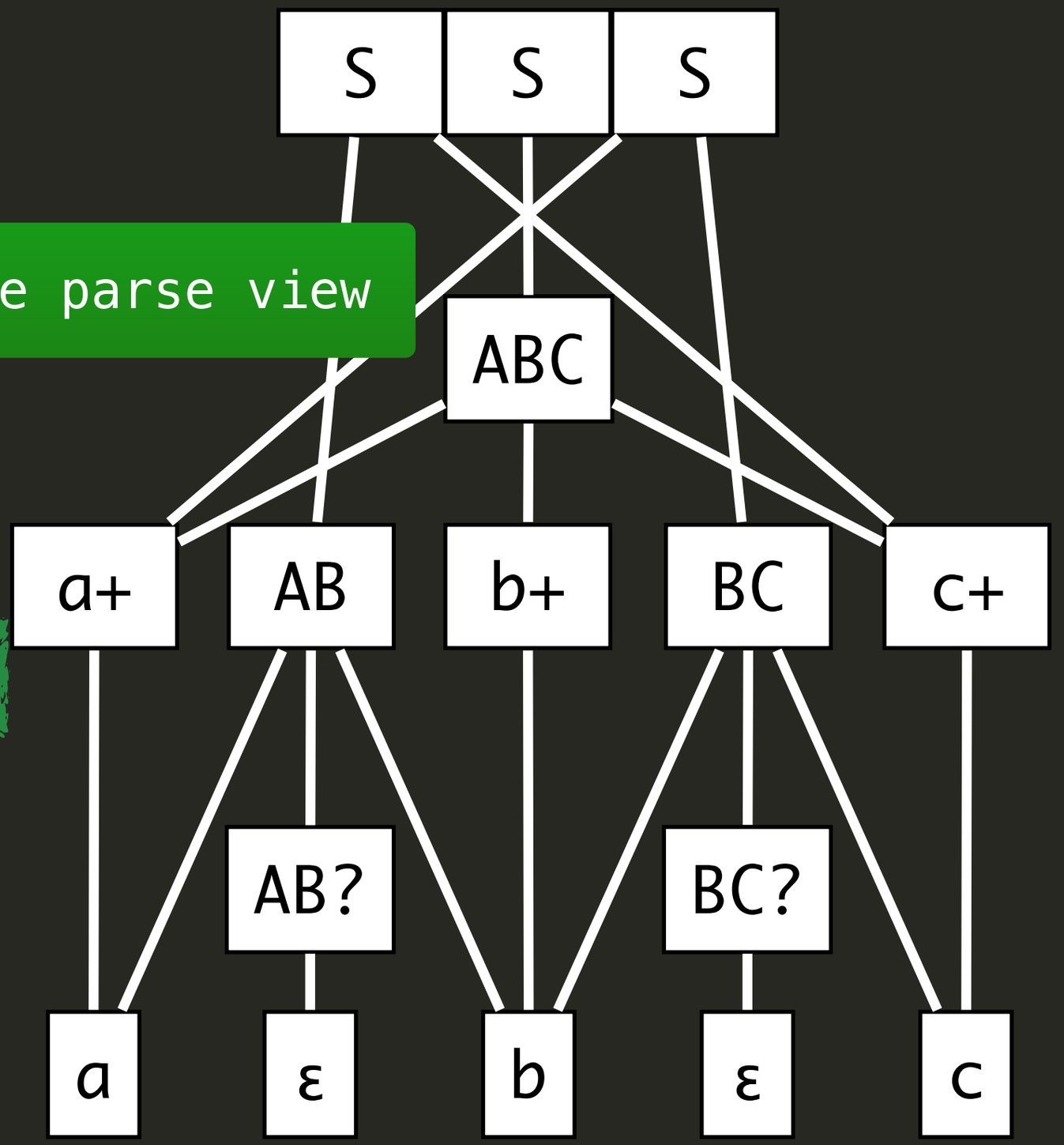
$$\{\ \langle e_1,(e_2,t_6,t_7)\rangle, \langle e_1,(e_4,t_4,e_6)\rangle, \langle e_1,(t_1,t_2,e_3)\rangle,$$
$$\langle e_2,(e_4,t_4,t_5)\rangle, \langle e_2,(t_1,t_2,e_5)\rangle, \langle e_3,(e_5,t_6,t_7)\rangle, \langle e_3,(t_3,t_4,e_6)\rangle,$$
$$\langle e_4,(t_1,t_2,t_3)\rangle, \langle e_5,(t_3,t_4,t_5)\rangle, \langle e_6,(t_5,t_6,t_7)\rangle\ \}$$

Boolean grammars

```
E ::= ABC
    & AB c+
    & a+ BC;

ABC ::= a+ b+ c+
AB  ::= a AB? b;
BC  ::= b AB? c;
```

nice parse view

context-sensitivity trick



The real problem:

transformation

The real problem:

- ✓ Manipulate “nice” views
  - ✓ let the rest keep up
- ✓ Merge views
  - ✓ disambiguate
- ✓ Modular transformations
  - ✓ temporary inconsistencies

- ✓ Many solutions
  - ✓ none satisfactory
- ✓ Standing challenge

State of the art [grammars]

XBGF

ΞBGF

SLEIR

. . .

http://grammarware.github.io/lab/

This paper

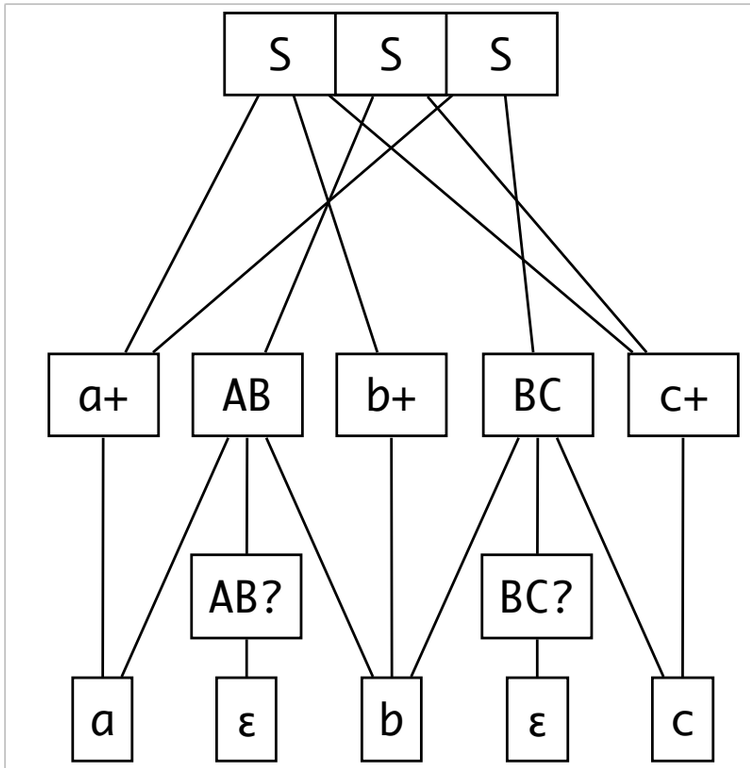
coupled

transformation

grammars

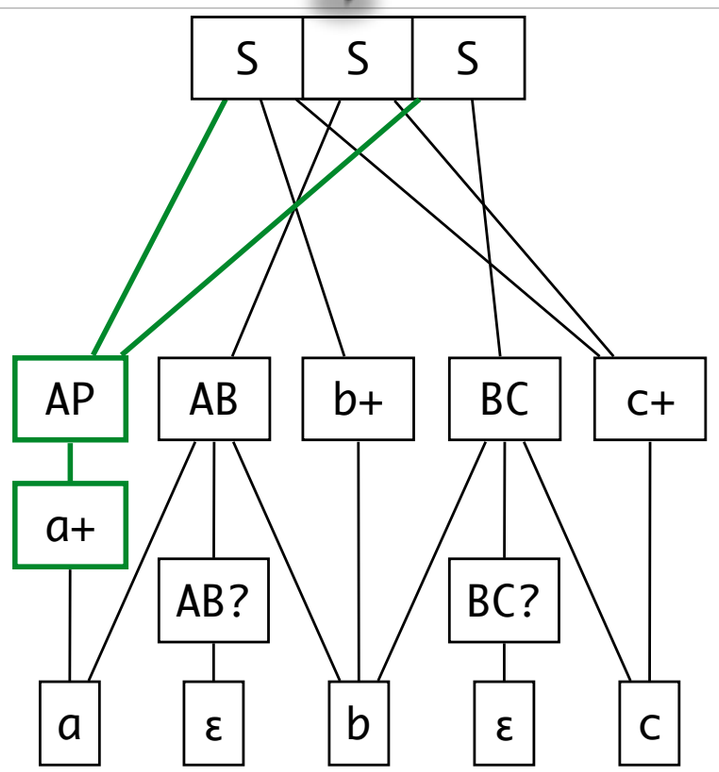
SPPFs

extract



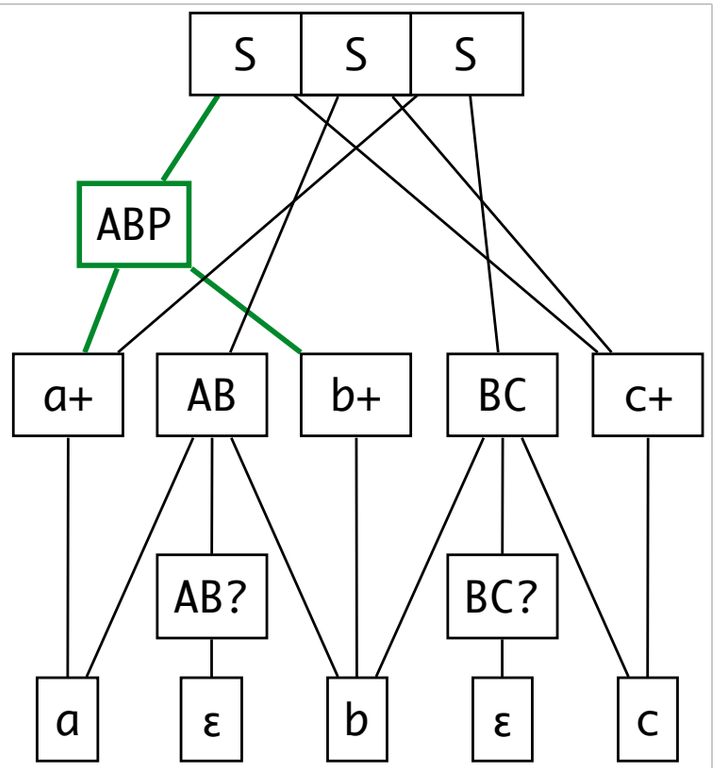

```
S  ::= a+ b+ c+ & AB c+ & a+ BC;
AB ::= a AB? b;
BC ::= b BC? c;
```

(a)




```
S  ::= AP b+ c+ & AB c+ & AP BC;
AB ::= a AB? b;
BC ::= b BC? c;
AP ::= a+;
```

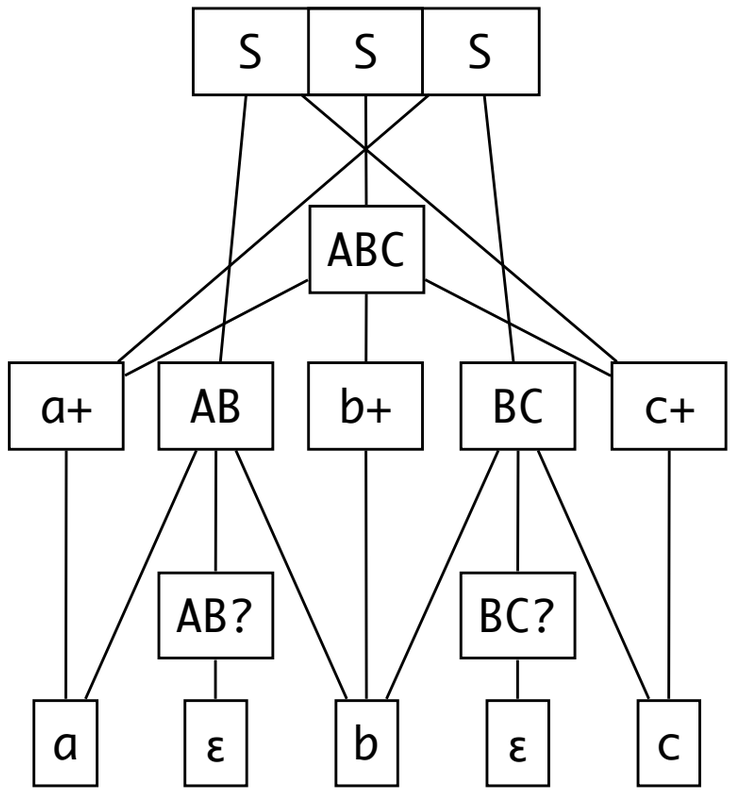
(b)




```
S   ::= ABP c+ & AB c+ & a+ BC;
AB  ::= a AB? b;
BC  ::= b BC? c;
ABP ::= a+ b+;
```

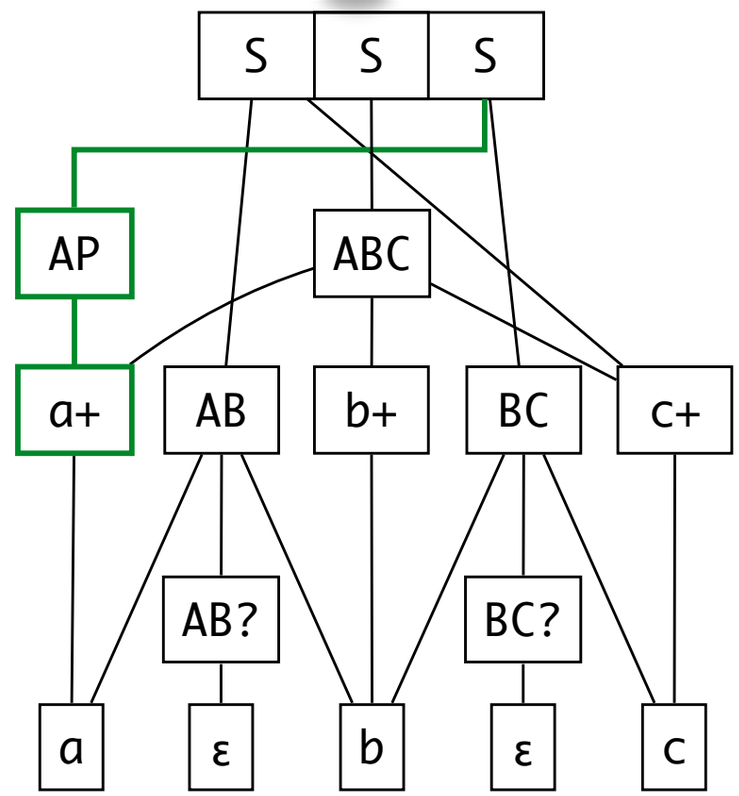
(c)

fold



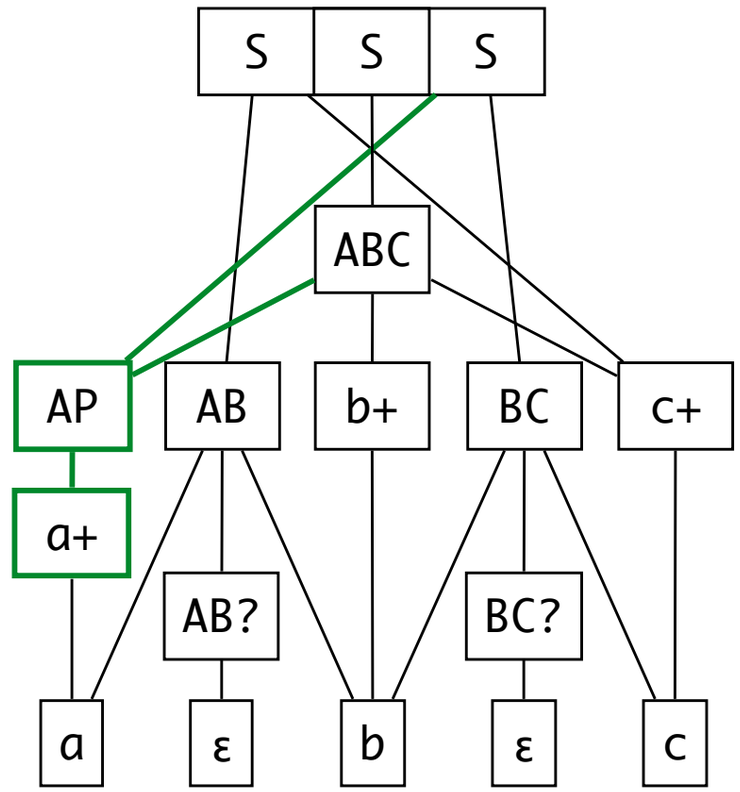

```
S   ::= ABC & AB c+ & a+ BC;
AB  ::= a AB? b;
BC  ::= b BC? c;
ABC ::= a+ b+ c+;
AP  ::= a+;
```

(a)




```
S   ::= ABC & AB c+ & AP BC;
AB  ::= a AB? b;
BC  ::= b BC? c;
ABC ::= a+ b+ c+;
AP  ::= a+;
```

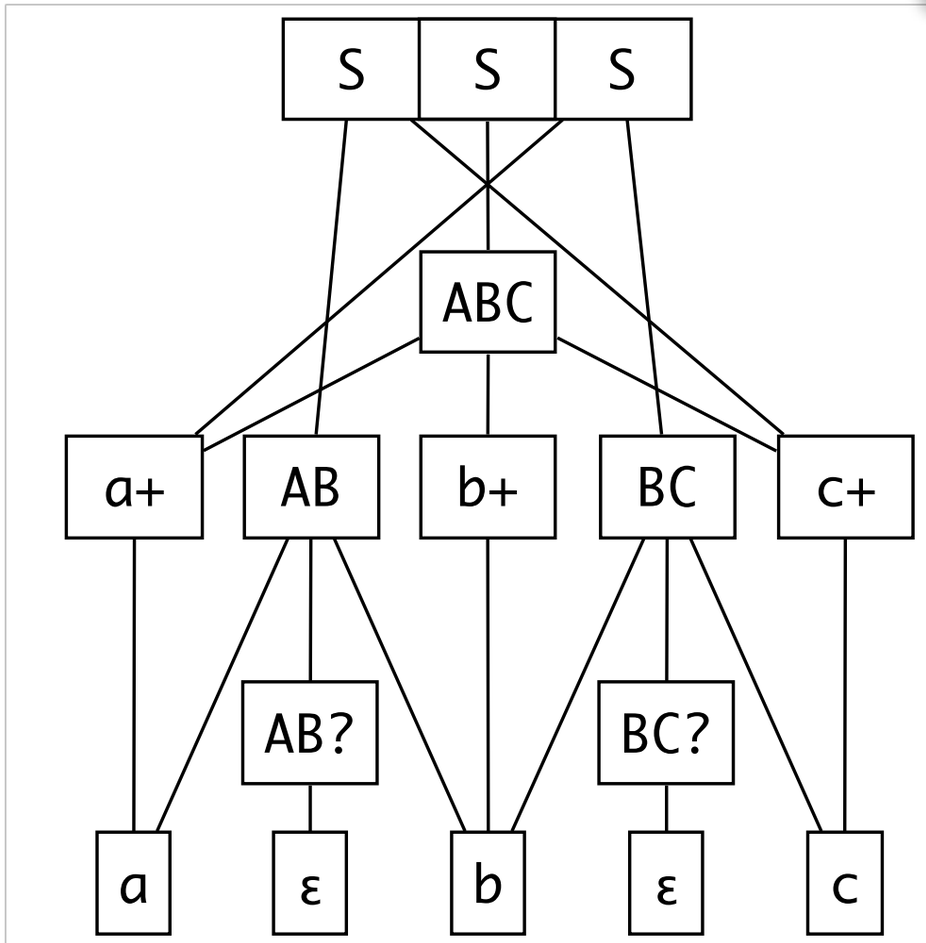
(b)




```
S   ::= ABC & AB c+ & AP BC;
AB  ::= a AB? b;
BC  ::= b BC? c;
ABC ::= AP b+ c+;
AP  ::= a+;
```

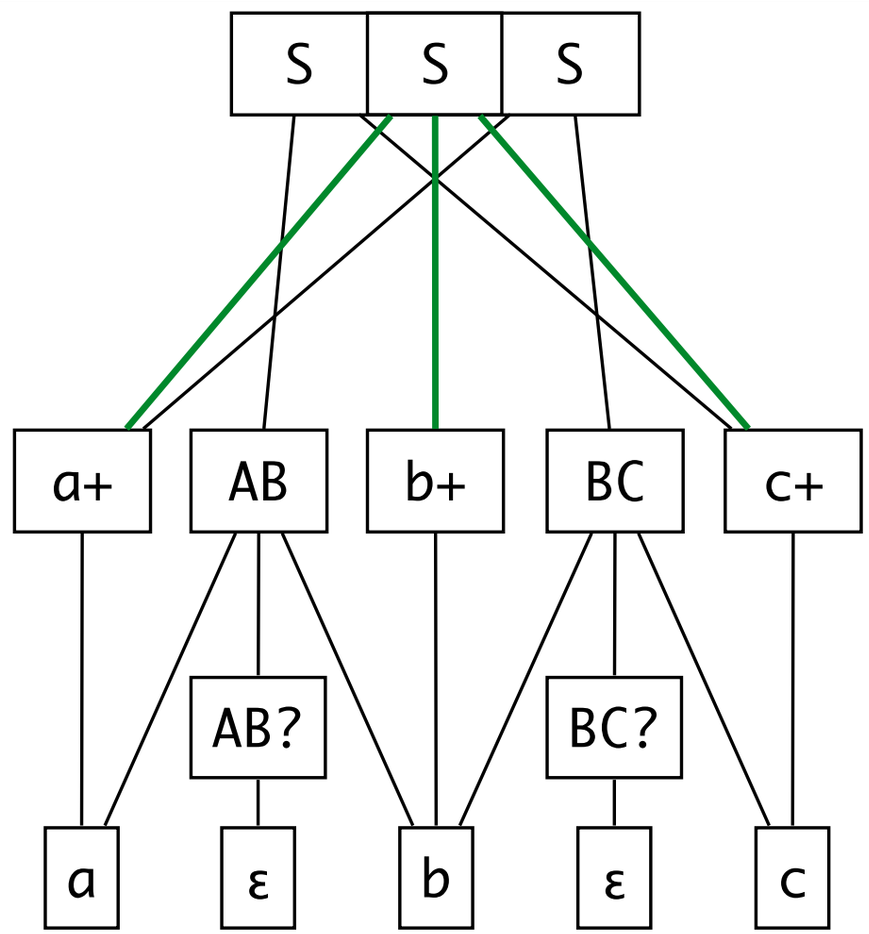
(c)

inline




```
S   ::= ABC & AB c+ & a+ BC;
AB  ::= a AB? b;
BC  ::= b BC? c;
ABC ::= a+ b+ c+;
```

(a)




```
S   ::= a+ b+ c+ & AB c+ & a+ BC;
AB  ::= a AB? b;
BC  ::= b BC? c;
```

(b)

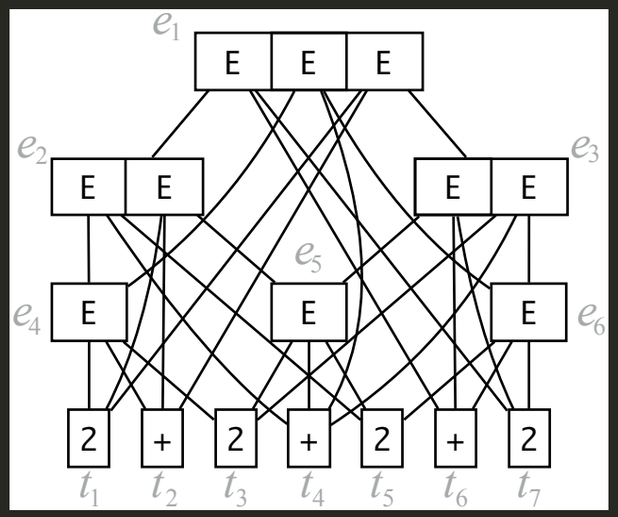
Solution (graphs)

Language	SPPFs	Examples
preserved	preserved	introduce, unlabel
preserved	refactored	fold, inline
extended	preserved	addV, define
extended	refactored	removeC, widen
reduced	preserved?	removeV, undefine
reduced	refactored	disappear
reduced	refactored?	addC, narrow
revised	refactored	permute, renameT
revised	refactored?	redefine, replace
revised	–	inject

Conclusion

- ✓ Programmably cotransform
  - ✓ string/term grammars
  - ✓ forest representations
- ✓ Applicable to grammars in a broad sense
  - ✓ views and models with uncertainty

- ✓ Still unclear
  - ✓ What is the best way?
  - ✓ Formalisation?



Questions?

Suggestions?