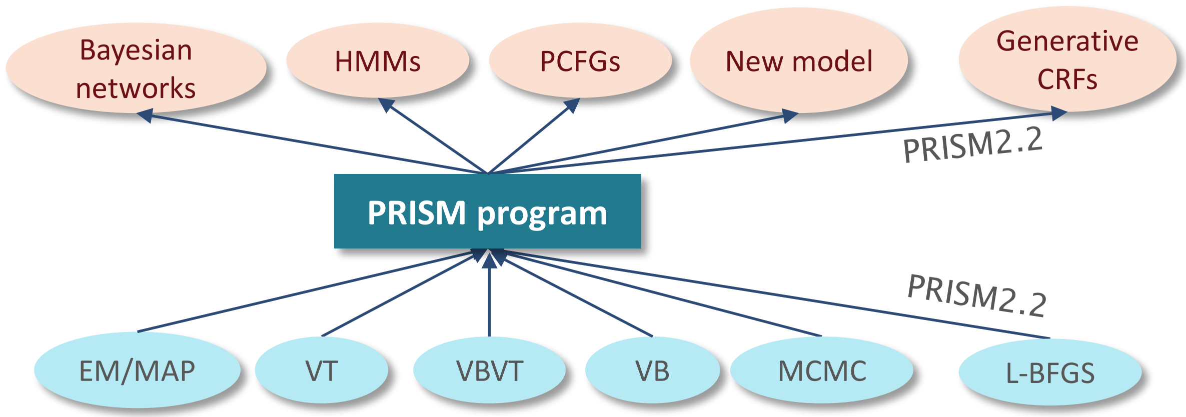
Goal recognition from incomplete action sequences by probabilistic grammars

Ryosuke Kojima & Taisuke Sato
Tokyo Institute of Technology

PRISM2.2

- PRISM [Sato et al. '97] (http://sato-www.cs.titech.ac.jp/prism/)
  - Probabilistic Prolog for machine and subsumes BN,HMM,PCFG,…



- PRISM2.2 has two new features
  - Learn and compute generative conditional random fields (G-CRFs)
    - logistic regression, linear-chain CRFs, CRF-CFGs
  - Can compute an infinite sum of probabilities
    - Markov chains, prefix and infix prob. in PCFGs

⬅ Today's topic

Probability computation in PRISM

- Goal ➔ expl. graph ➔ probs

PCFG$_1$
S ➔ a:0.5 | b:0.3 | S S:0.2

```
values(s,[[a],[b],[s,s]],
      set@[0.5,0.3,0.2]).

pcfg(L):- pcfg([s],L,[]).
pcfg([A|R],L0,L2):-
   ( nonterminal(A) ->
       msw(A,RHS),
       pcfg(RHS,L0,L1)
   ;   L0=[A|L1] ),
   pcfg(R,L1,L2).
pcfg([],L,L).
```

PCFG program

?- prob(pcfg([s],[a,b],[]),P)

Tabled search

G-IO comp. by DP

0.2

```
pcfg([s],[a,b],[])
  ⇔ pcfg([s,s],[a,b],[]) & pcfg([],[],[]) & msw(s,[s,s])
pcfg([s,s],[a,b],[])
  ⇔ pcfg([a],[a,b],[b]) & pcfg([s],[b],[]) & msw(s,[a])
pcfg([s],[b],[])
  ⇔ pcfg([b],[b],[]) & pcfg([],[],[]) & msw(s,[b])
pcfg([b],[b],[]) ⇔ pcfg([],[],[])
pcfg([],[],[])
pcfg([a],[a,b],[b]) ⇔ pcfg([],[b],[b])
pcfg([],[b],[b])
```

0.5

0.3

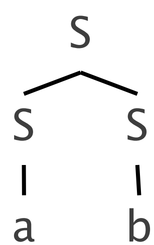
P = 0.5x0.3x0.2 = 0.03

Probabilities are automatically learned from data by learn/1 in PRISM

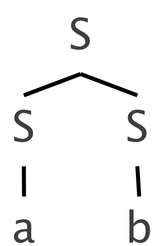
New feature: infinite sum
–Prefix probability computation

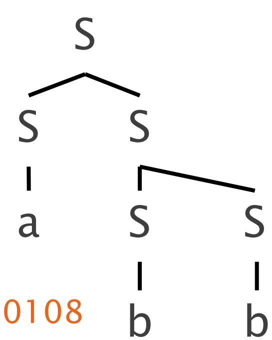
- Prefix u: uw is a sentence for some w
- $P_{refix}(u) = \Sigma_{uw:setntence} P(uw)$
- $PCFG_1$ (probabilistic context free grammar)

$$S \rightarrow a{:}0.5 \mid b{:}0.3 \mid S\,S{:}0.2$$

$$P_{cfg}([a,b]) = P\Big(\ $$


$$\Big) = \overset{0.2}{P(S \rightarrow S\ S)}\ \overset{0.5}{P(S \rightarrow a)}\ \overset{0.3}{P(S \rightarrow b)}$$

$$P_{refix}([a,b]) = P\Big(\ $$


$$\Big) + P\Big(\ $$


$$\Big) + \cdots$$

0.03 0.0108

Cyclic explanation graph

- Goal ➔ expl. graph ➔ SCCs ➔ linear eqs ➔ probs

PCFG$_2$
S ➔ a:0.4 | b:0.3 | S S:0.2 | S:0.1

```
values(s,[[a],[b],[s,s],[s]],
        set@[0.4,0.3,0.2,0.1]).
pre_pcfg(L):-
    pre_pcfg([s],L,[]).
pre_pcfg([A|R],L0,L2):-
   ( nonterminal(A) ➔
       msw(A,RHS),
       pre_pcfg(RHS,L0,L1)
   ;   L0=[A|L1] ),
   ( L1=[] ➔ L2=[]
   ; pre_pcfg(R,L1,L2) ).
pre_pcfg([],L,L).
```

prefix parser

prefix

?- lin_prob(pre_pcfg([s],[a,b],[]),P)

Tabled search

Cyclic dependency!

```
pre_pcfg([s],[a,b],[])
   ⇔ pre_pcfg([s,s],[a,b],[]) & msw(s,[s,s])
    v pre_pcfg([s],[a,b],[]) & msw(s,[s])
pre_pcfg([s,s],[a,b],[])
   ⇔ pre_pcfg([a],[a,b],[b]) & pre_pcfg([s],[b],[]) &
      msw(s,[a])
    v pre_pcfg([s,s],[a,b],[]) & msw(s,[s,s])
    v pre_pcfg([s],[a,b],[b]) & pre_pcfg([s],[b],[]) &
      msw(s,[s])
    v pre_pcfg([s],[a,b],[]) & msw(s,[s])
 …
```

Solving linear equations

P = 0.05

SCCs (strongly connected components)

- SCCs are partially ordered ➔ DP possible

SCC

SCC

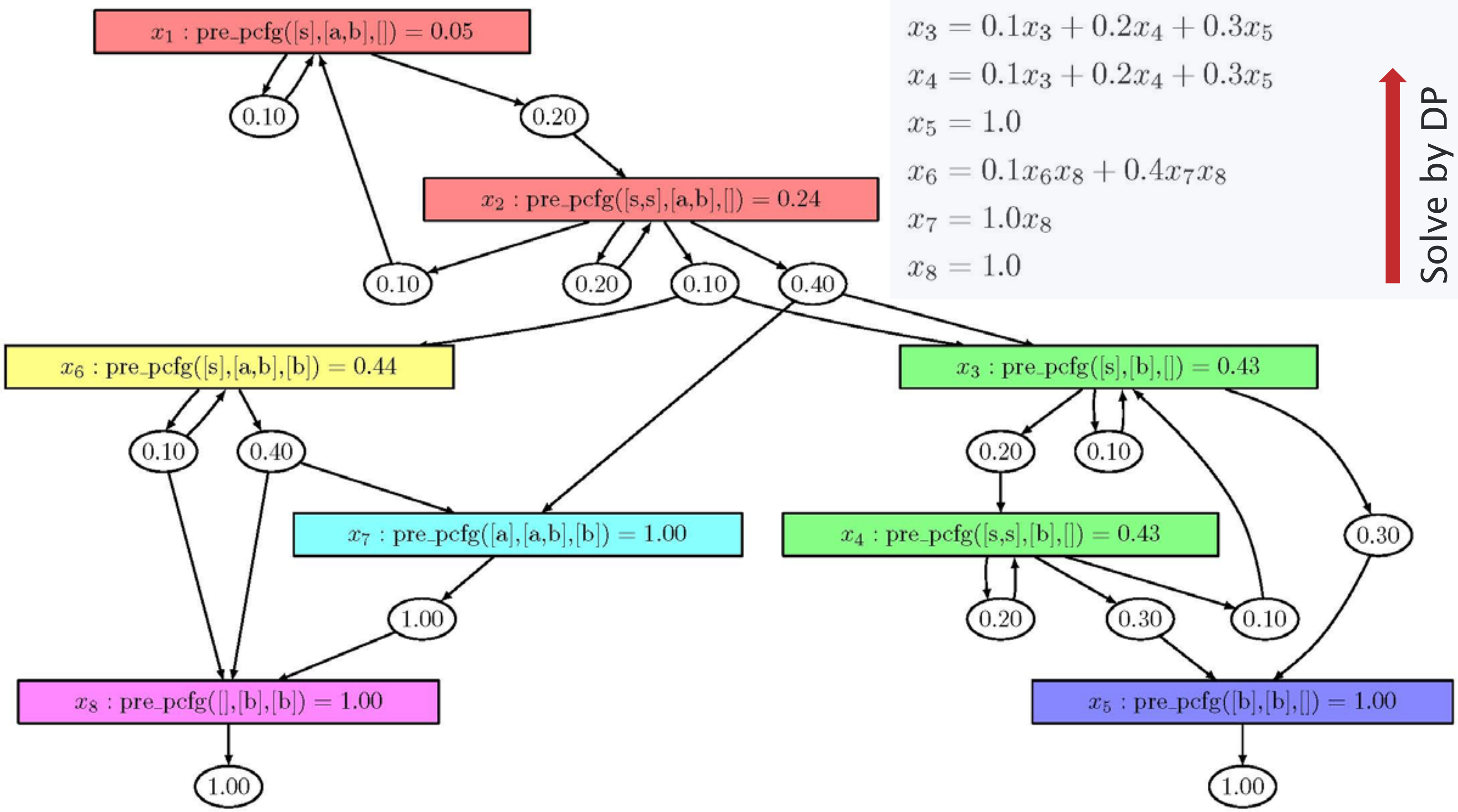
SCC

pre_pcfg([s],[a,b],[])
  ⇔ pre_pcfg([s,s],[a,b],[]) & msw(s,[s,s]) v pre_pcfg([s],[a,b],[]) & msw(s,[s])
pre_pcfg([s,s],[a,b],[])
  ⇔ pre_pcfg([a],[a,b],[b]) & pre_pcfg([s],[b],[]) & msw(s,[a])
   v pre_pcfg([s,s],[a,b],[]) & msw(s,[s,s])
   v pre_pcfg([s],[a,b],[b]) & pre_pcfg([s],[b],[]) & msw(s,[s])
   v pre_pcfg([s],[a,b],[]) & msw(s,[s])
pre_pcfg([s],[b],[])
  ⇔ pre_pcfg([b],[b],[]) & msw(s,[b]) v pre_pcfg([s,s],[b],[]) & msw(s,[s,s])
   v pre_pcfg([s],[b],[]) & msw(s,[s])
pre_pcfg([s,s],[b],[])
  ⇔ pre_pcfg([b],[b],[]) & msw(s,[b]) v pre_pcfg([s,s],[b],[]) & msw(s,[s,s])
   v pre_pcfg([s],[b],[]) & msw(s,[s])
pre_pcfg([b],[b],[])
pre_pcfg([s],[a,b],[b])
  ⇔ pre_pcfg([a],[a,b],[b]) & pre_pcfg([],[b],[b]) & msw(s,[a])
   v pre_pcfg([s],[a,b],[b]) & pre_pcfg([],[b],[b]) & msw(s,[s])
pre_pcfg([a],[a,b],[b]) ⇔ pre_pcfg([],[b],[b])
pre_pcfg([],[b],[b])

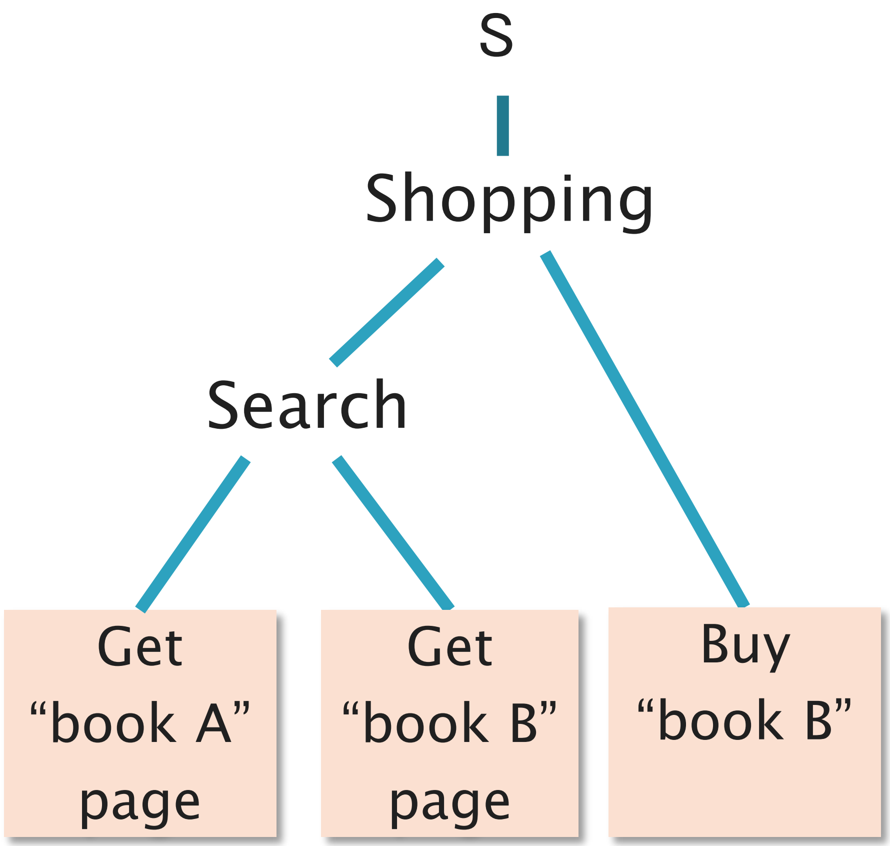
A set of probability equations

$$
\begin{aligned}
x_1 &= 0.1x_1 + 0.2x_2 \\
x_2 &= 0.1x_1 + 0.2x_2 + 0.1x_3x_6 + 0.4x_3x_7 \\
x_3 &= 0.1x_3 + 0.2x_4 + 0.3x_5 \\
x_4 &= 0.1x_3 + 0.2x_4 + 0.3x_5 \\
x_5 &= 1.0 \\
x_6 &= 0.1x_6x_8 + 0.4x_7x_8 \\
x_7 &= 1.0x_8 \\
x_8 &= 1.0
\end{aligned}
$$

Solve by DP



Plan recognition [Kautz+ 91, Vilain 90]



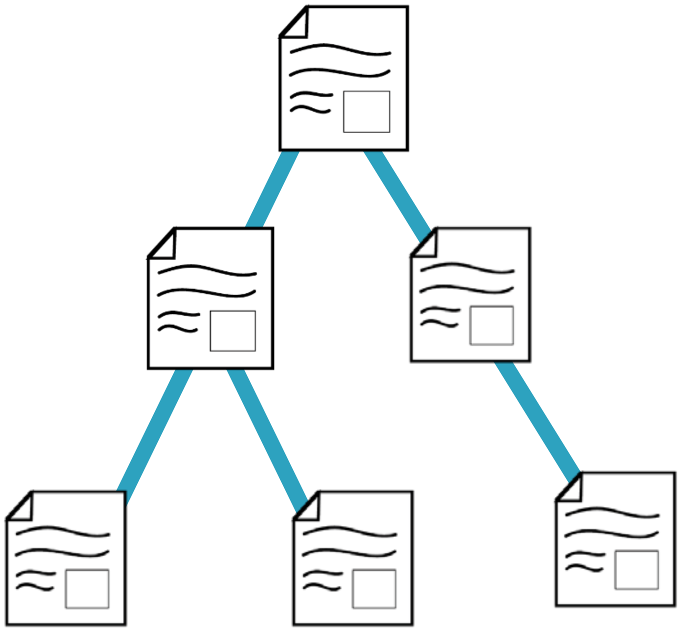
Plan	Parse tree in a PCFG
action	word
action seq	word seq
completed action seq	sentence

Web surfing
– capturing the user's intension online

up	: climb up
down	: down
sibling	: visit sibling page
revisit	: visit same page
move	: others

We observe an action sequence as a prefix in a PCFG and infer its underlying plan as a most-likely nonterminal using prefix probability

Web site



path:

/en/publication/index.html

Experiment

- Web data :
  - access log (action sequences) from the Internet Traffic Archive (NASA(2014), ClarkNet(4523), U of S(652))
- Task :
  - to classify prefixes of access log data into five plans (survey,news,…)
  - Four methods (HMM,Prefix, LR,SVM) used

    generative discriminative

Gold standard

- Five plans (intentions) detected by clustering
  - Clustering access log data from the Internet Traffic Archive (NASA, ClarkNet, U of S) by a mixture of PCFGs (CFG rules common, parameters different) yields five clusters
- We write 102 CFG rules (32 NTs)

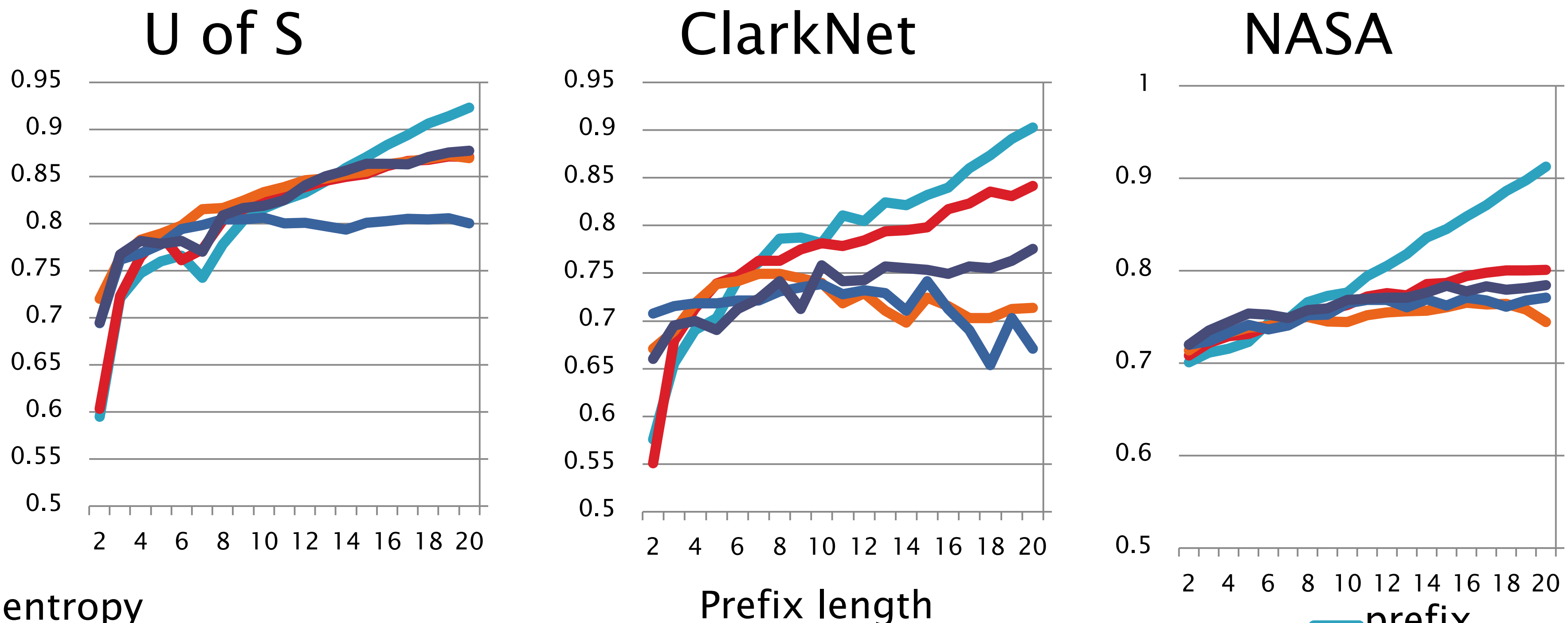
  S → Survey [0.2], S → News [0.4],⋯

  UpDown →Up,Down [0.3]

  UpDown →Up, SameLayer, Down [0.6]

  Up →Up, up [0.2],…
- We determine the gold standard
  - Access log data paired with a category inferred by the Viterbi algorithm using a mixture of PCFGs (parameters are estimated from access log data as sentences)

Classification accuracy



entropy

5.12×10^4 2.77×10^5 3.14×10^6

(PCFG's entropy: $-\sum_t p(t) \log p(t)$ of PCFG[Chi+99])

The prefix method performs better when the prefix is long

Conclusion and announcement

- PRISM2.2 allows cyclic explanation graphs and can compute probabilities of PCFGs' prefixes by solving a set of probability equations.
- We applied prefix probability computation to plan recognition from access log data in web sites.
- The prefix method outperformed HMM, LR, (two types of) SVMs when prefix length is long.
- The pre-release of PRISM2.2 is available from http://sato-www.cs.titech.ac.jp/prism/