Nested Words and Automata

Alvin George

Indian Institute of Science

2019 Nov 29

Overview

- Motivation
- Nested Words
- Operations on Nested Words
- Nested Words Applications

Appeal of Regular Languages

- Well-understood expressiveness: multiple characterizations
  - Deterministic/nondeterministic/alternating finite automata
  - Regular expressions
  - Monadic second order logic of linear order
  - Syntactic congruences

- Regular languages are effectively closed under many operations
  - Union, intersection, complement, conactenation, Kleene-*, homomorphisms…

- Algorithms for decision problems
  - Membership
  - Determinization and minimization
  - Language emptiness (single-source graph reachability)
  - Language inclusion, language equivalence …

Checking Structured Programs

- Control-flow requires stack, so (abstracted) program P defines a context-free language
- Algorithms exist for checking regular specifications against context-free models
  - Emptiness of pushdown automata is solvable
  - Product of a regular language and a context-free language is context-free
- But, checking context-free spec against a context-free model is undecidable!
  - Context-free languages are not closed under intersection
  - Inclusion as well as emptiness of intersection undecidable
- Existing software model checkers: pushdown models (Boolean programs) and regular specifications

Are Context-free Specs Interesting?

- Classical Hoare-style pre/post conditions
  - If p holds when procedure A is invoked, q holds upon return
  - Total correctness: every invocation of A terminates
  - Integral part of emerging standard JML
- Stack inspection properties (security/access control)
  - If setuuid bit is being set, root must be in call stack
- Interprocedural data-flow analysis
- All these need matching of calls with returns, or finding unmatched calls
  - Recall: Language of words over [,] such that brackets are well matched is not regular, but context-free

Checking Context-free Specs

- Many tools exist for checking specific properties
  - Security research on stack inspection properties
  - Annotating programs with asserts and local variables
  - Inter-procedural data-flow analysis algorithms

- What's common to checkable properties?
  - Both program P and spec S have their own stacks, but the two stacks are synchronized

- As a generator, program should expose the matching structure of calls and returns

Solution: Nested words and theory of regular languages over nested words

Program Executions as Nested Words

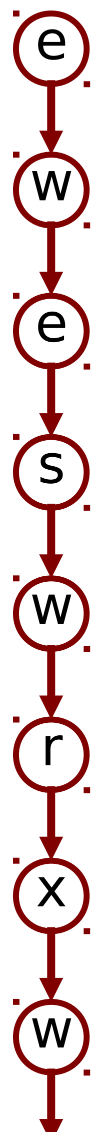
Program

```
global int x;
main() {
  x = 3;
  if P   x = 1 ;
  ....
}

bool P () {
 local int y=0;
  x = y;
  return (x==0);
}
```

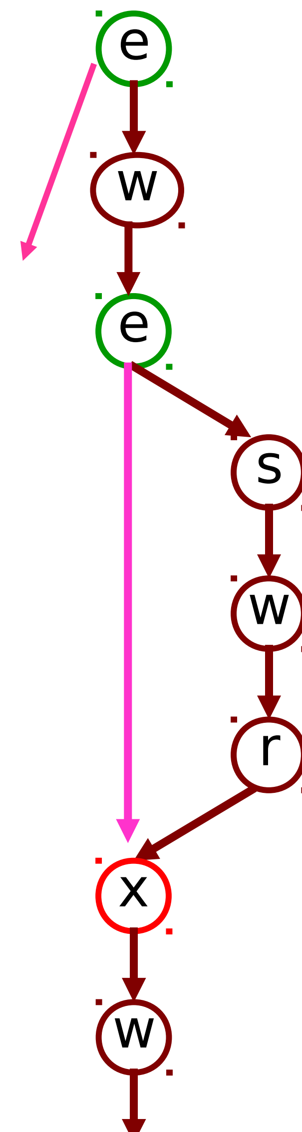
If a procedure writes to x, it must later read it

An execution as a word



Symbols:
w : write x
r : read x
e: enter
x: exit
s : other

An execution as a nested word



Summary edges from calls to returns

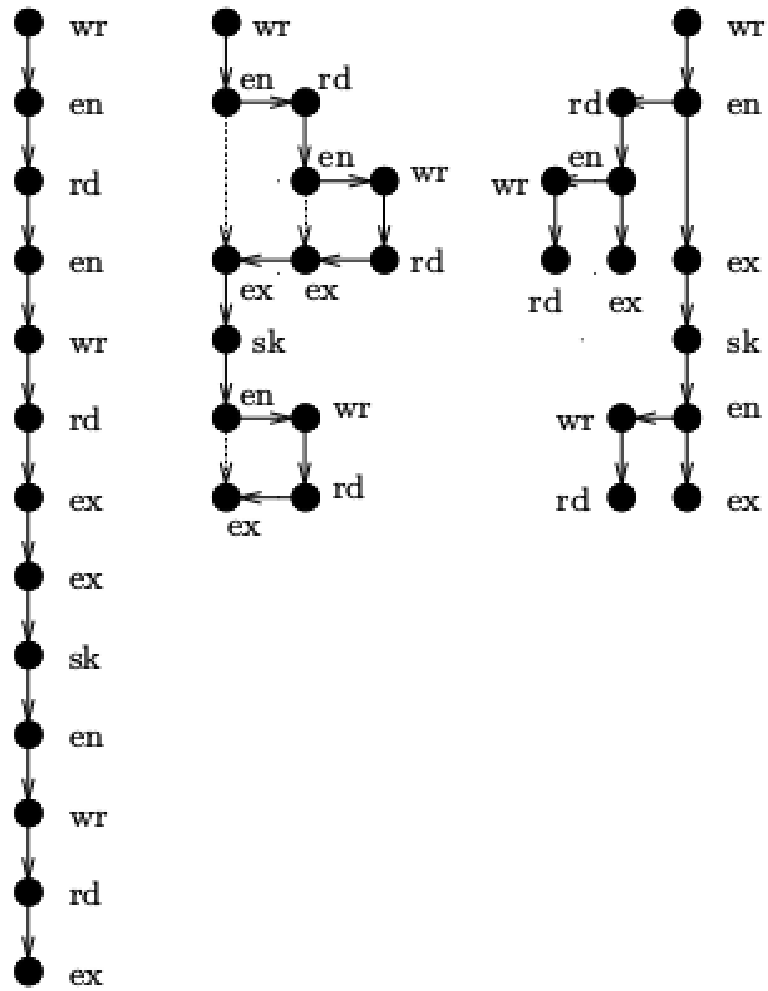
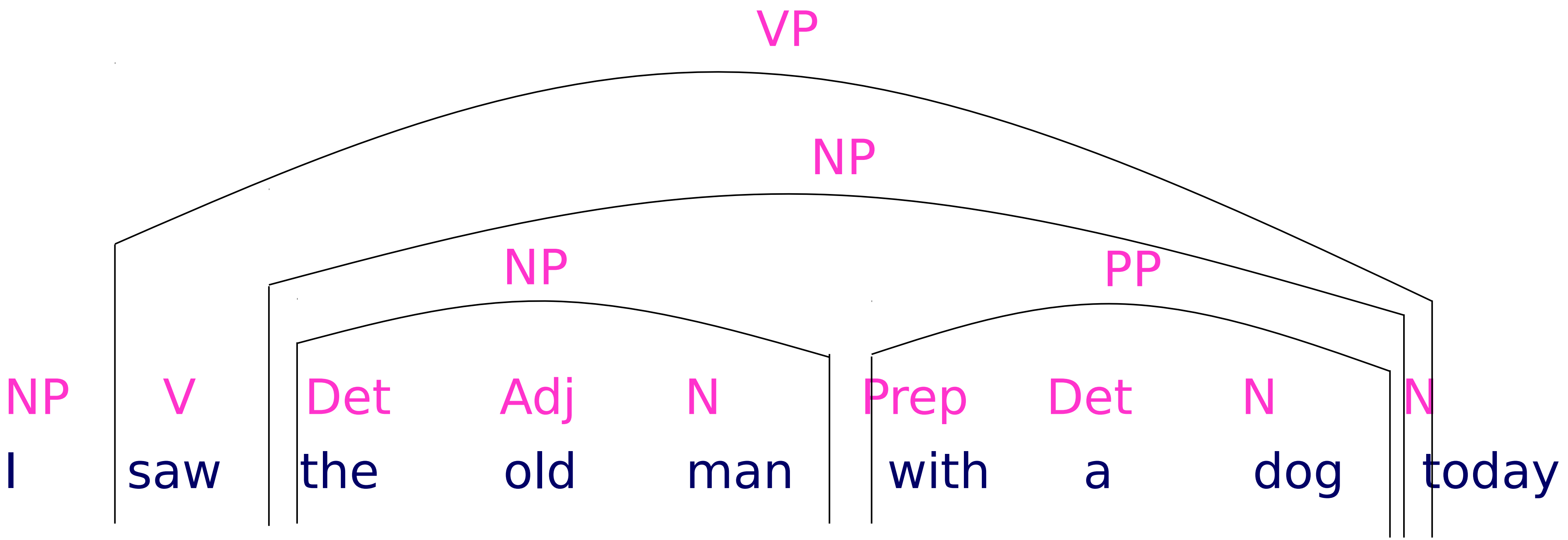


Fig. 1. Execution as a word, as a nested word, and as a tree

Linguistic Annotated Data



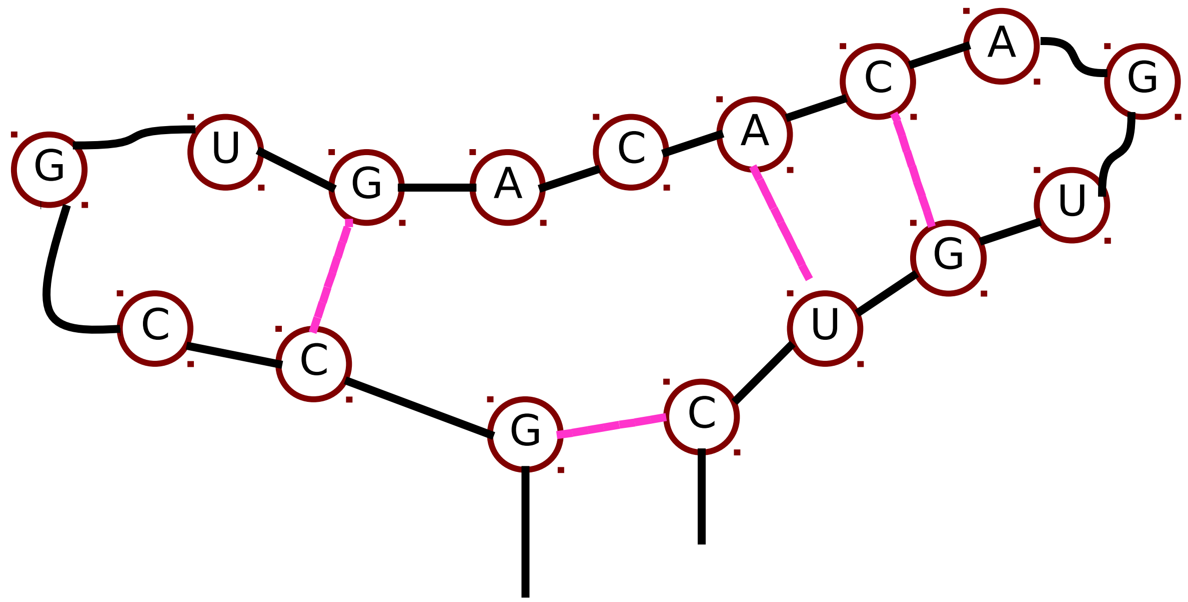
Linguistic data stored as annotated sentences (eg. Penn Treebank)

Sample query: Find nouns that follow a verb which is a child of a verb phrase

RNA as a Nested Word

Primary structure: Linear sequence of nucleotides (A, C, G, U)

Secondary structure: Hydrogen bonds between complementary nucleotides (A-U, G-C, G-U)



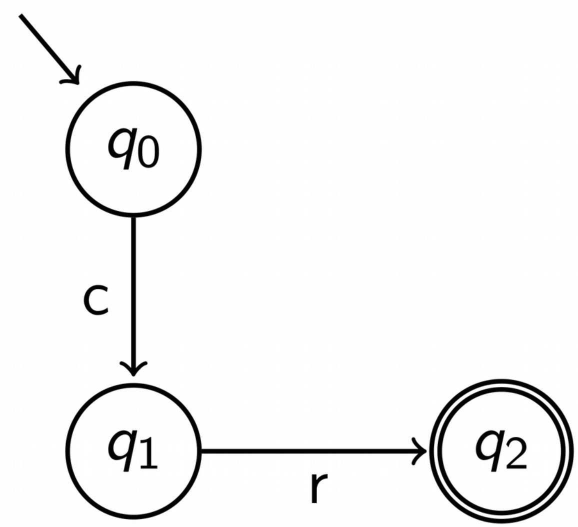
In literature, this is modeled as trees.

Algorithmic question: Find similarity between RNAs using edit distances

Regular language

```
procedure foo()
{
    return;
}
```

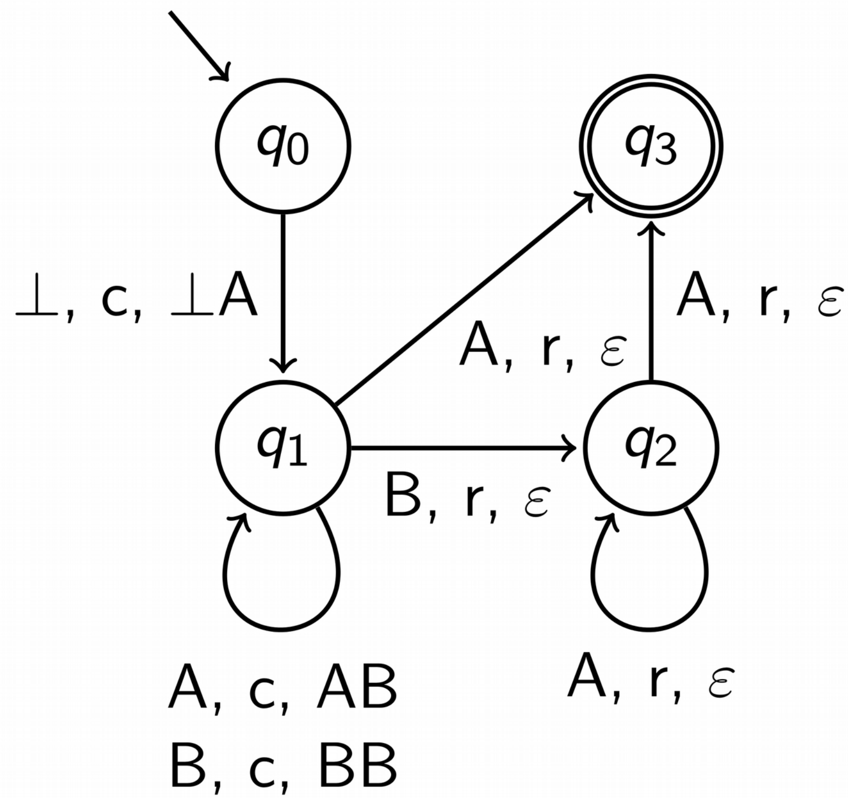
$$\mathcal{L}_1 = \{\text{c r}\}$$



(det.) Context-free language

```
procedure bar()
{
    if (*)
        call bar();
    return;
}
```

$$\mathcal{L}_2 = \{c^n\ r^n \mid n > 0\}$$



Comparison

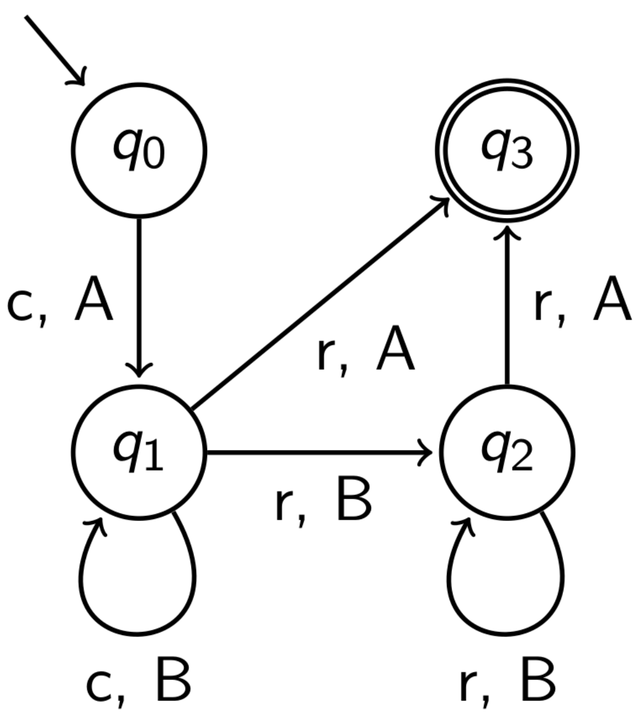
☺ ☹ ☹ ☹	**regular**	**context-free**
comparison of numbers	constants	two variables
closure	all standard properties	not under intersection and complementation
decidability	all standard problems	intersection, inclusion, equivalence undecidable
determinize	powerset construction	not possible

Question: Is there some class of languages in between that is more expressive than regular languages, but keeps their nice properties?

Answer (Alur & Madhusudan 2004): yes, at least in some sense

$\mathcal{L}_2$ as VPL

Consider again $\mathcal{L}_2 = \{\mathrm{c}^n\ \mathrm{r}^n \mid n > 0\}$. We construct a VPA for $\mathcal{L}_2$.



Partitioning:

$\Sigma_i = \emptyset$, $\Sigma_c = \{c\}$, $\Sigma_r = \{r\}$

$$\delta_c = \{\ (q_0, c, A, q_1),\ (q_1, c, B, q_1)\ \}$$

$$\delta_r = \{\ (q_1, r, A, q_3),\ (q_1, r, B, q_2),\ (q_2, r, A, q_3),\ (q_2, r, B, q_2)\ \}$$

From VPAs to NWAs

- main differences between VPAs and PDAs:
  - closed under determinism
  - partitioning of the alphabet
  - very limited use of the stack
- Do we really need the stack?
  (Alur & Madhusudan 2006): no, with some further treatment of the input $\rightarrow$ *nested words* (NWs)
- automaton model: *nested word automata* (NWAs)
- *nested word languages* (NWLs) and VPLs have same power
  $\rightarrow$ NWAs $\preceq$ deterministic PDAs
- main idea: call and return symbols are matched in the input

Nested words

A relation $\rightsquigarrow \subset \{-\infty, 1, 2, \ldots, \ell\} \times \{1, 2, \ldots, \ell, \infty\}$ of length $\ell \geq 0$ is a *matching relation* if the following holds:

- I if $i \rightsquigarrow j$, then $i < j$ (monotone)
- II if $i_1 \rightsquigarrow j$ and $i_2 \rightsquigarrow j$, then $i_1 = i_2$ (left-unique)
  if $i \rightsquigarrow j_1$ and $i \rightsquigarrow j_1$, then $j_1 = j_2$ (right-unique)
- III if $i_1 \rightsquigarrow j_1$ and $i_2 \rightsquigarrow j_2$, then we have not $i_1 < i_2 < j_1 < j_2$ (well nested)

Explanation:

- I not **r c**, not reflexive
- II not **c c r**, not **c r r**
- III not **c c r r**

> ex post note: $(-\infty, \infty) \notin \rightsquigarrow$,
> $\pm\infty$ excluded from uniqueness

Nested words

A relation $\rightsquigarrow \subset \{-\infty, 1, 2, \ldots, \ell\} \times \{1, 2, \ldots, \ell, \infty\}$ of length $\ell \geq 0$ is a *matching relation* if the following holds:

I if $i \rightsquigarrow j$, then $i < j$ (monotone)

II if $i_1 \rightsquigarrow j$ and $i_2 \rightsquigarrow j$, then $i_1 = i_2$ (left-unique)
if $i \rightsquigarrow j_1$ and $i \rightsquigarrow j_1$, then $j_1 = j_2$ (right-unique)

III if $i_1 \rightsquigarrow j_1$ and $i_2 \rightsquigarrow j_2$, then we have not $i_1 < i_2 < j_1 < j_2$ (well nested)

If $i \rightsquigarrow j$, i is a *call position* and j is a *return position*. All the rest is an *internal position*. If $i \neq -\infty$ and $j \neq \infty$, they are *well-matched*, otherwise *pending*. $e \in \rightsquigarrow$ is a *nesting edge*.

A *nested word* n over Σ is a pair $(a_1 \cdots a_\ell, \rightsquigarrow)$, where $a_i \in \Sigma$ and $\rightsquigarrow$ is a matching relation of length ℓ.

Well nested sequences

A sequence of symbols is *well nested* if calls and returns are matched without crossing, i.e., for any different call-return-pairs (c_i, r_i), (c_j, r_j), $c_i < c_j < r_i < r_j$ is forbidden.

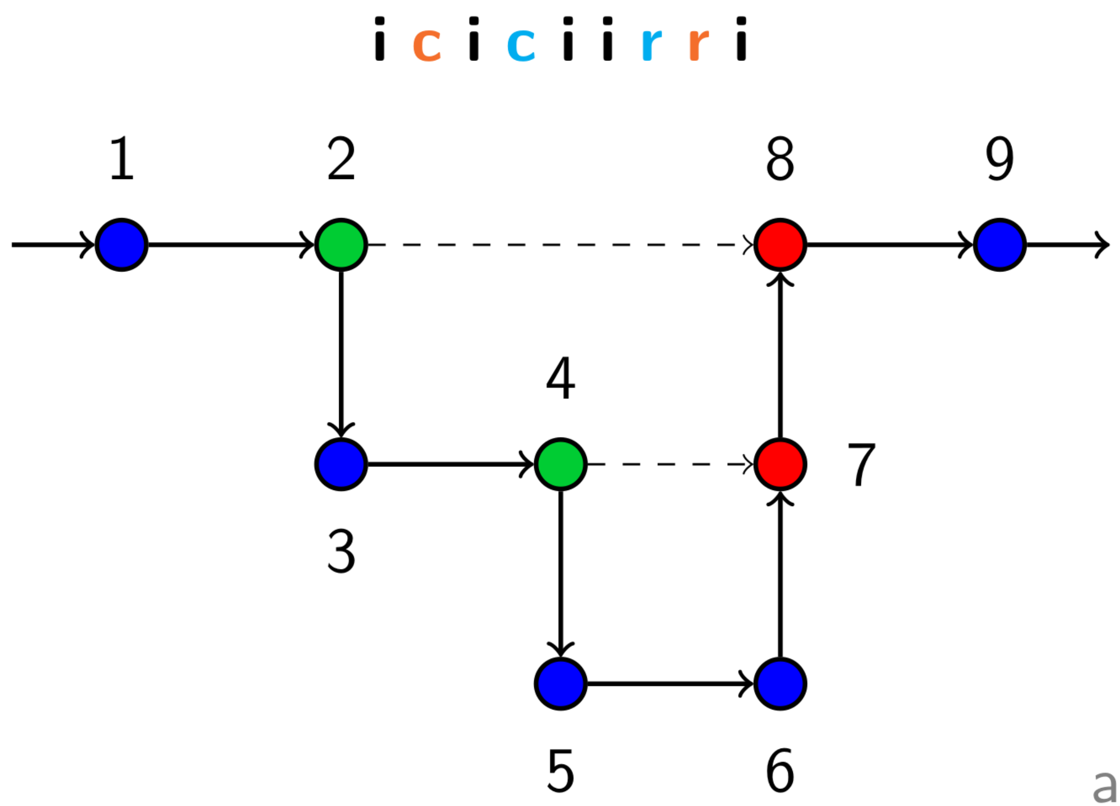
Examples:

i c i c i i r r i

r c r r c i c i

Note: Every sequence has a unique well nesting.

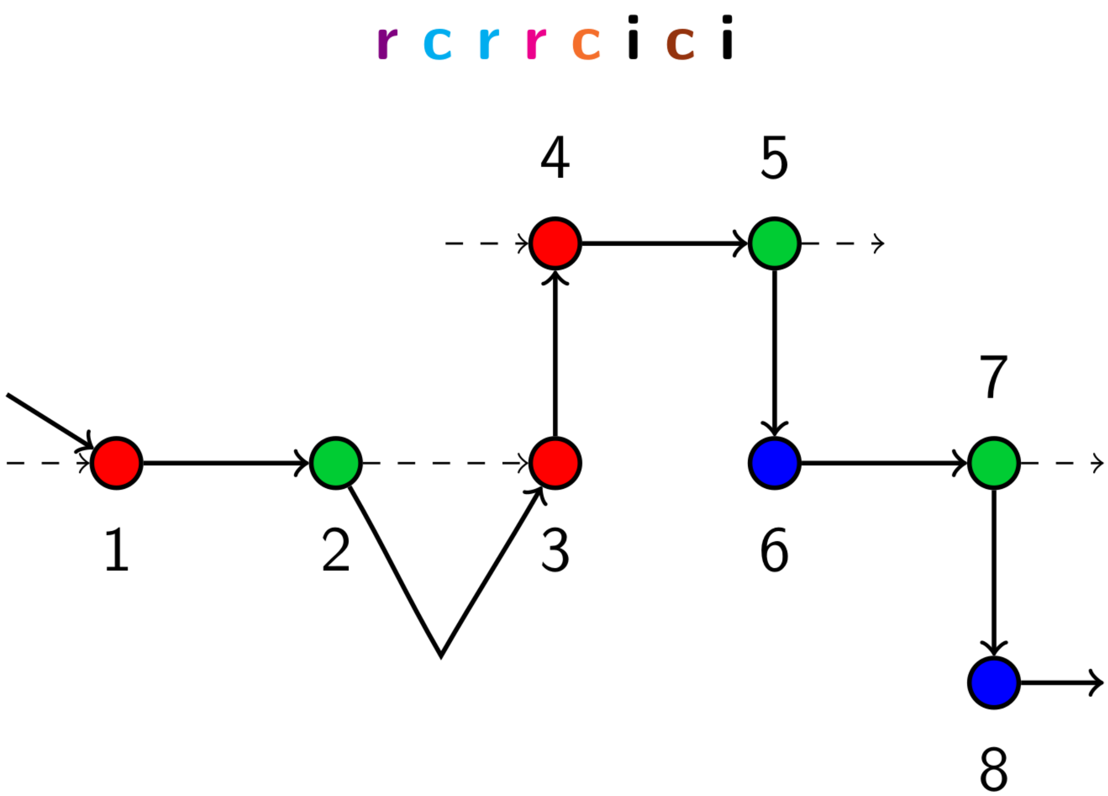
Example 1



adapted from [1]

Here: $2 \rightsquigarrow 8$, $4 \rightsquigarrow 7$ and the whole word is well-matched.

Example 2



adapted from [1]

Here: $-\infty \rightsquigarrow 1$, $2 \rightsquigarrow 3$, $-\infty \rightsquigarrow 4, 5 \rightsquigarrow \infty, 7 \rightsquigarrow \infty$ and only $2 \rightsquigarrow 3$ is well-matched.

Program Executions as Nested Words

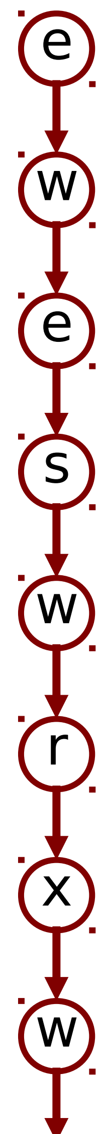
Program

```
global int x;
main() {
  x = 3;
  if P   x = 1 ;
  ….
}

bool P () {
 local int y=0;
  x = y;
  return (x==0);
}
```

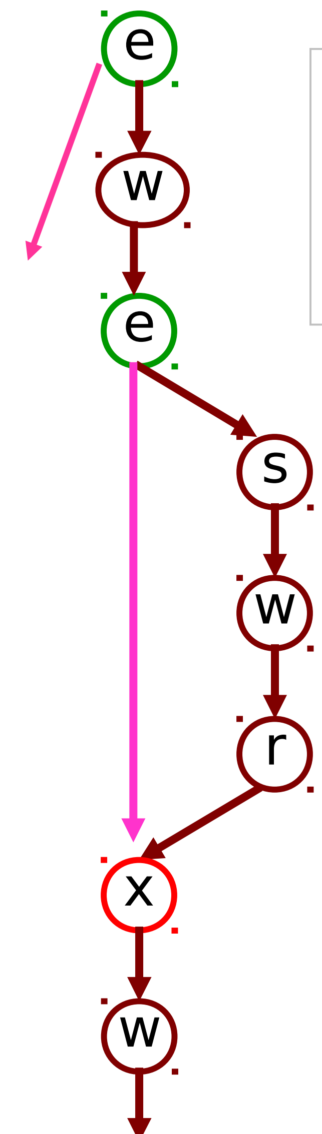
If a procedure writes
to x, it must later read it

An execution
as a word

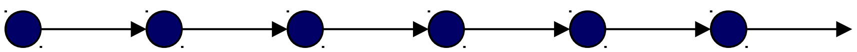


Symbols:
w : write x
r : read x
e: enter
x: exit
s : other

An execution
as a nested word

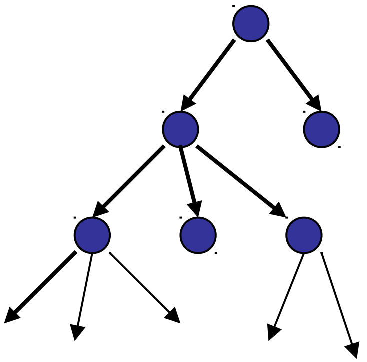


Summary
edges
from calls
to returns



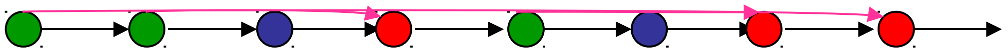
Words:

Data with linear order



(Unordered) Trees:

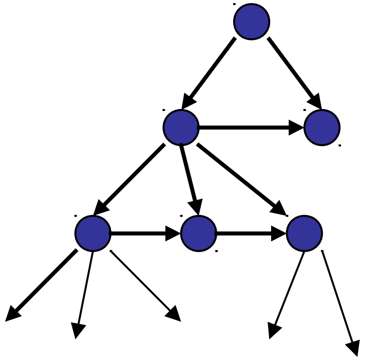
Data with hierarchical order



Nested Words (AM06):

Data with linear order +

Nesting edges



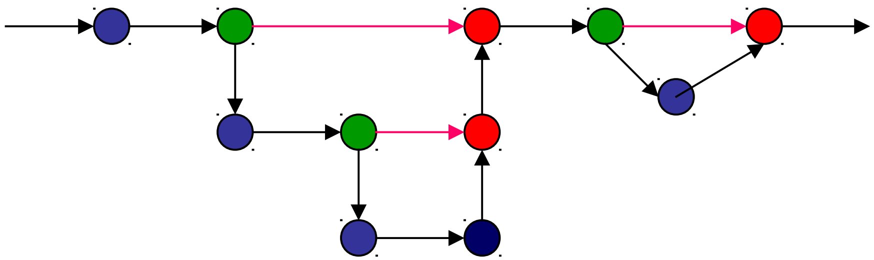
Ordered Trees/Hedges:

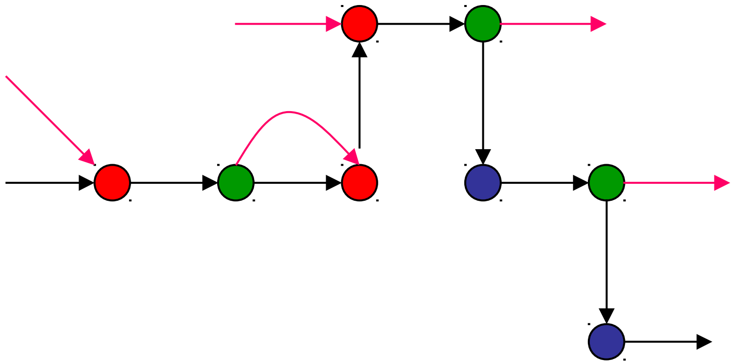
Data with hierarchical order +

Linear order on siblings

Nested Shape:

- Linear sequence + Non-crossing nesting edges
- Nesting edges can be pending, Sequence can be infinite





Positions classified as:

- Call positions: both linear and hierarchical outgoing edges
- Return positions: both linear and hierarchical incoming edges
- Internal positions: otherwise

Nested word:

Nested shape + Positions labeled with symbols in Σ

Concatenation of Nested Words

Given two nested words
nw1 = (w1 , ν1) and nw2 = (w2 , ν2),
of lengths k1 and k2 , respectively,

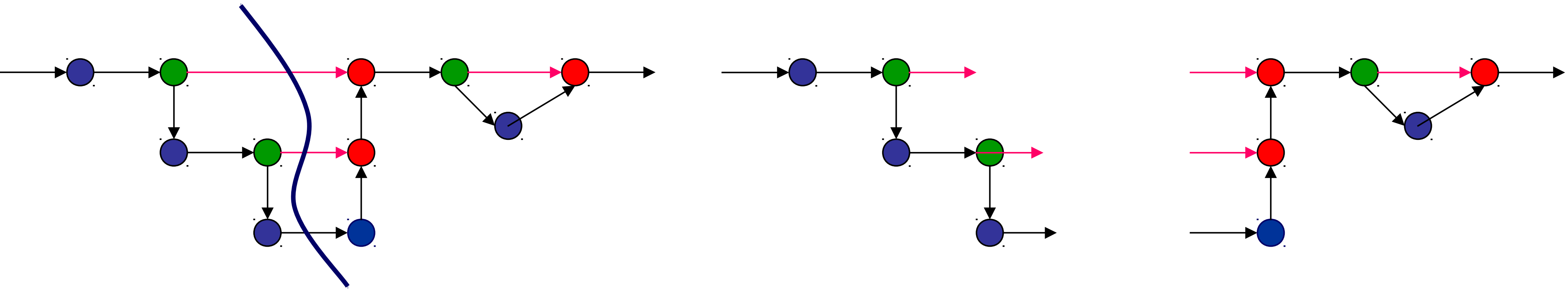
the concatenation of nw1 and nw2 is the nested word

nw1 .nw2 = (w1 .w2 , ν) of length k1 +k2 ,
where ν is the nested relation

$$\nu_1 \cup \{(k_1 + i,\ k_1 + j) \mid (i, j) \in \nu_2\}.$$

Word operations:

Prefixes, suffixes, concatenation, reverse



Insertion for nested words

A context is a pair (nw, i)
where nw is a nested word of length k, and $0 \leq i \leq k$.

Given a context (nw, i), for nw = ($a_1 \dots a_k$, ν),
and a nested word nw′, withnw′ = (w′, ν′),

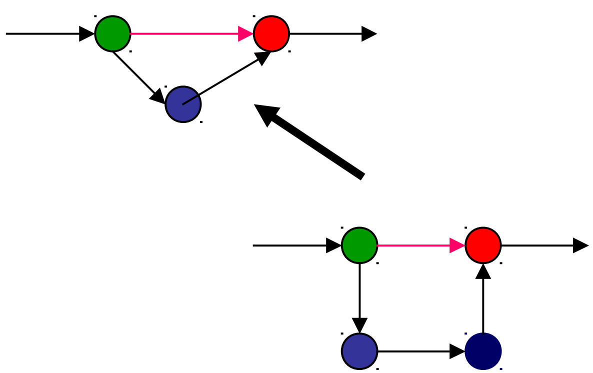
(nw, i) ⊕ nw′ is the nested word obtained by inserting the nested word nw′ at position i in nw.

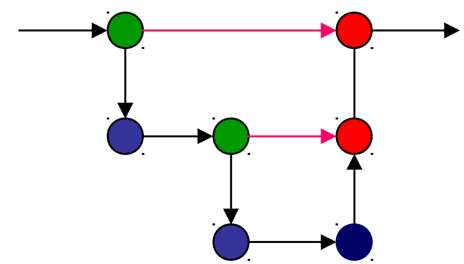
More precisely, (nw, i) ⊕ nw′ is the
nested word ($a_1 \dots a_i . w' . a_{i+1} \dots a_k$, ν″), where
$$\nu'' = \{(\pi_1(j), \pi_1(j')) \mid (j, j') \in \nu\} \cup \{(\pi_2(j), \pi_2(j')) \mid (j, j') \in \nu'\}$$

where $\pi_1(j) = j$, if $j \leq i$,
$= |w'| + j$ otherwise,
and $\pi_2(j) = i + j$

Tree operations:

- Inserting/deleting well-matched words
- Well-matched: no pending calls/returns





Word Encoding

- The tagged alphabet $\hat{\Sigma}$ to be the set $\Sigma \cup <\Sigma \cup \Sigma>$. Formally,
- we define the mapping nw_w : NW $(\Sigma) \rightarrow \hat{\Sigma}*$

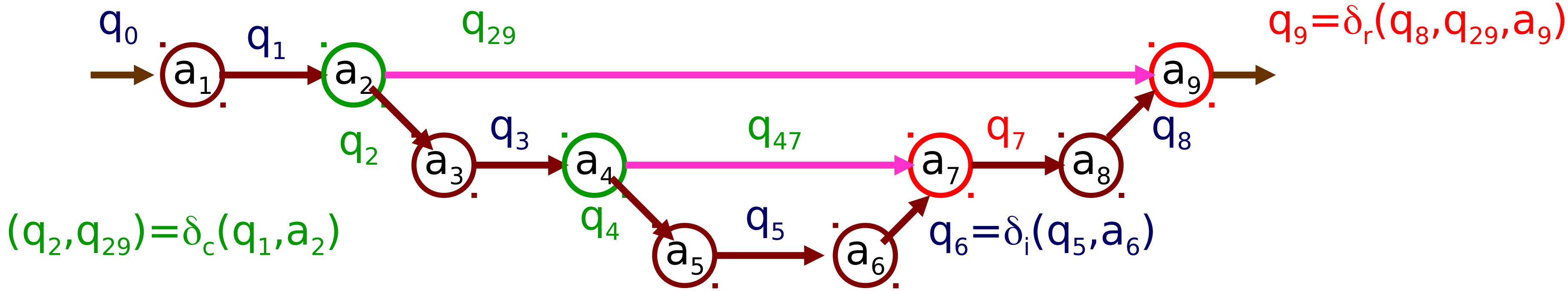
Definition of NWAs

$\mathcal{A} = \langle Q, q_0, Q_f, P, p_0, P_f, \delta_i, \delta_c, \delta_r \rangle$ over alphabet Σ

- Q finite set of *linear* states,
- $q_0 \in Q$ initial *linear* state,
- $Q_f \subseteq Q$ set of *linear* final states,
- P finite set of *hierarchical* states,
- $p_0 \in Q$ initial *hierarchical* state,
- $P_f \subseteq P$ set of *hierarchical* final states,
- $\delta_i \subseteq Q \times \Sigma \to Q$ internal transition function,
- $\delta_c \subseteq Q \times \Sigma \to Q \times P$ call transition function,
- $\delta_r \subseteq Q \times P \times \Sigma \to Q$ return transition function

acceptance via both Q_f and P_f
as VPAs: at return implicitly go to hierarchical state before matching call

Nested Word Automata (NWA)

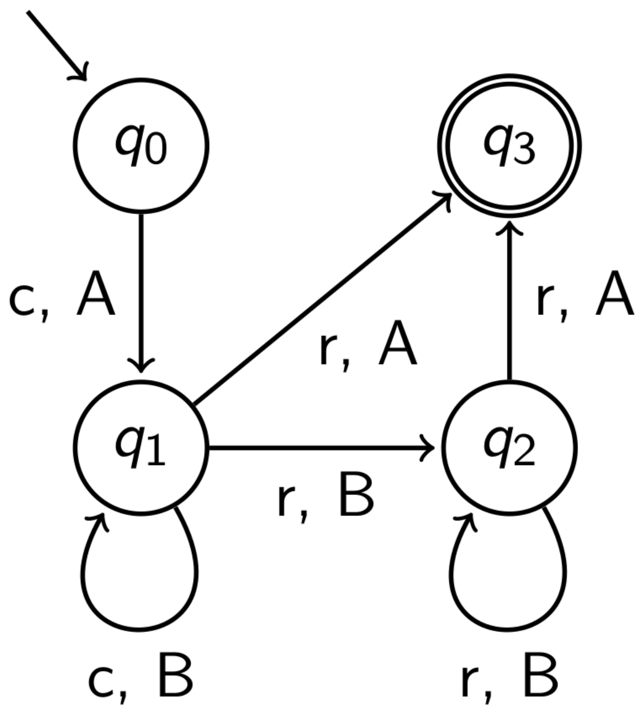


- States Q, initial state q_0, final states F
- Reads the word from left to right labeling edges with states
- Transition function:
  - **δ_c** : Q x Σ -> Q x Q (for call positions)
  - **δ_i** : Q x Σ -> Q (for internal positions)
  - **δ_r** : Q x Q x Σ -> Q (for return positions)
- Nested word is accepted if the run ends in a final state

$\mathcal{L}_2$ as NWA

Consider again $\mathcal{L}_2 = \{\mathsf{c}^n\ \mathsf{r}^n \mid n > 0\}$.

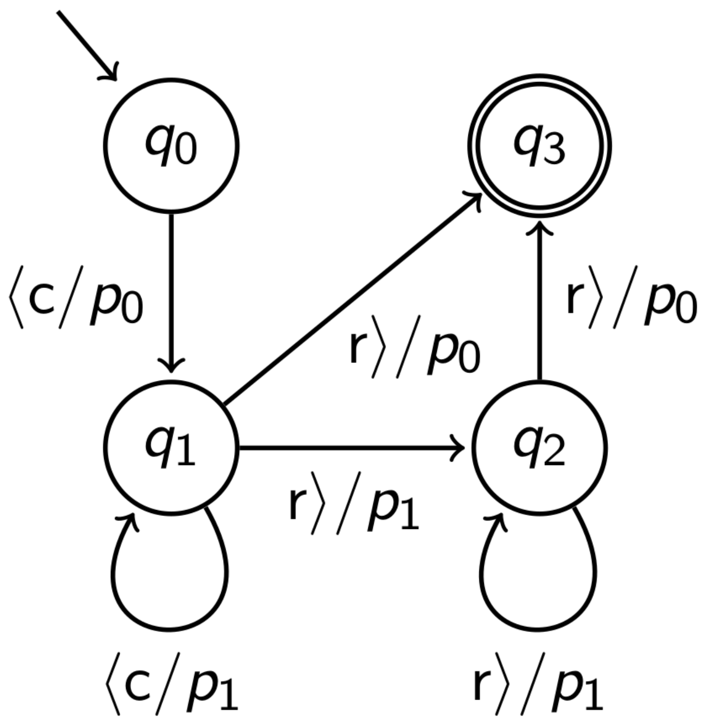
We construct an NWA for $\mathcal{L}_2' := \{(\langle\mathsf{c})^n\ (\mathsf{r}\rangle)^n \mid n > 0\}$.



$\mathcal{L}_2$ as NWA

Consider again $\mathcal{L}_2 = \{\mathsf{c}^n\ \mathsf{r}^n \mid n > 0\}$.

We construct an NWA for $\mathcal{L}_2' := \{(\langle \mathsf{c})^n\ (\mathsf{r}\rangle)^n \mid n > 0\}$.



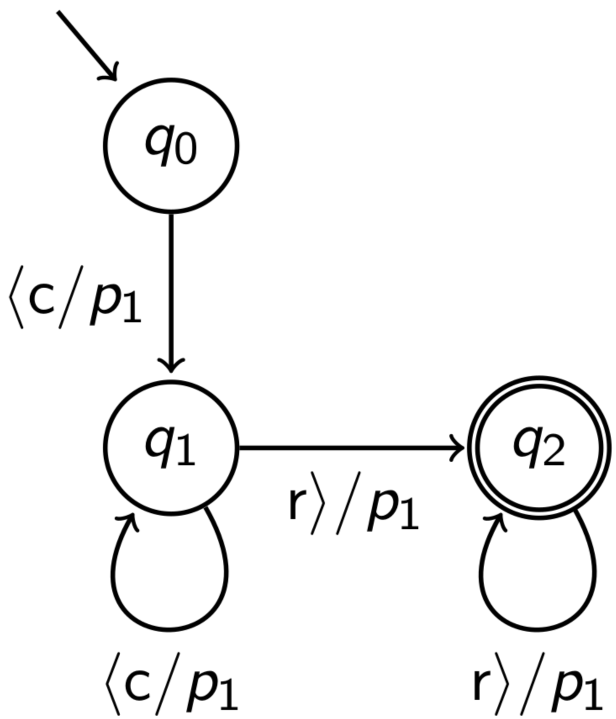
$P = \{p_0, p_1\}$, $P_f \subseteq \{p_0\}$

$\mathcal{L}_2$ as NWA

Consider again $\mathcal{L}_2 = \{\mathsf{c}^n\ \mathsf{r}^n \mid n > 0\}$.

We construct an NWA for $\mathcal{L}_2' := \{(\langle\mathsf{c})^n\ (\mathsf{r}\rangle)^n \mid n > 0\}$.

We can also use hierarchical states for acceptance.



$P = \{p_0, p_1\}$, $P_f = \{p_0\}$

Remarks

- no stack anymore, but structure on the input word
- nondeterministic NWAs: $Q_0 \subseteq Q$, $P_0 \subseteq P$, δ
  possibly exponentially more states for deterministic NWAs
- not all sets of NWs acceptable by NWAs
  $\{(\langle a)^n (b\rangle)^n \mid n > 0\}$ vs. $\{a^n b^n \mid n > 0\}$

Comparison of properties

	DFA	DNWA	PDA	DPDA
pre-/suffix	✓	✓	✓	✓
$\cup, \cdot, *$	✓	✓	✓	✗
complement	✓	✓	✗	✓
$\cap$	✓	✓	✗	✗
emptiness	NLOGSPACE	PTIME	PTIME	PTIME
equivalence	NLOGSPACE	PTIME	undecidable	decidable
inclusion	NLOGSPACE	PTIME	undecidable	undecidable

Note: Equivalence and inclusion problem are EXPTIME-complete for nondeterministic NWAs.
Implication: determinization $\in \Omega(\text{EXPTIME})$ if at all possible

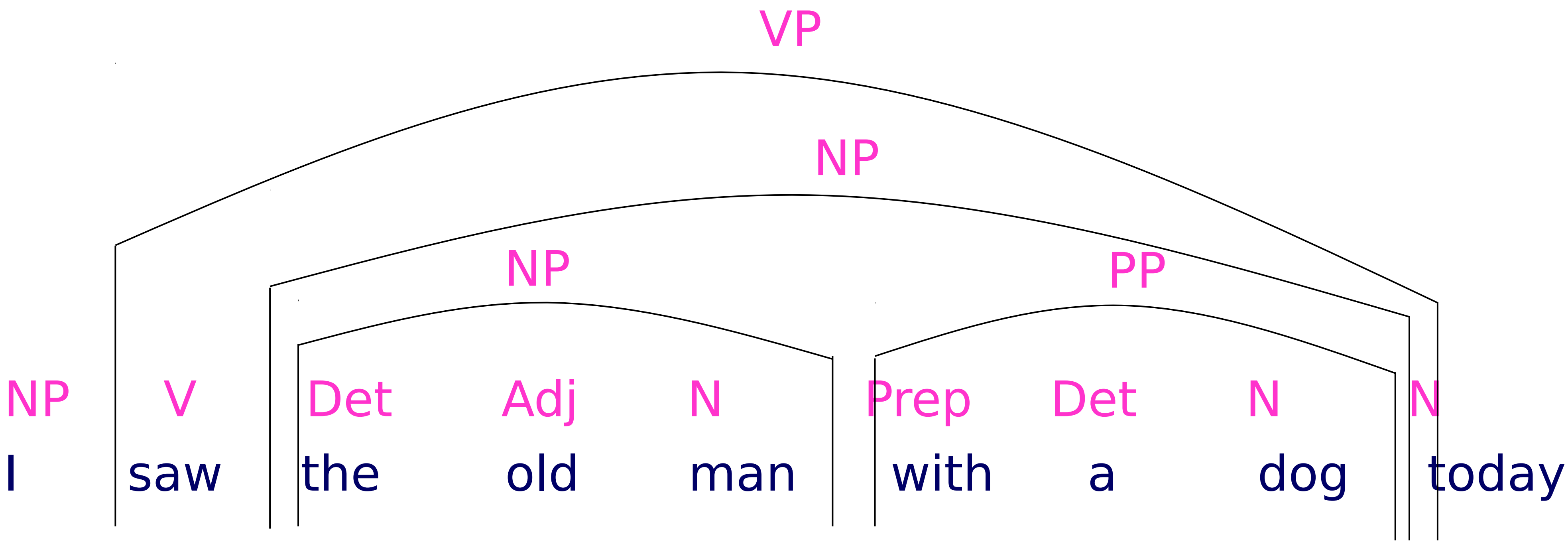
Regular Languages of Nested Words

- A set of nested words is regular if there is a finite-state NWA that accepts it

- Nondeterministic automata over nested words
  - Transition function: δ_c: $Q x \Sigma \to 2^{QxQ}$, δ_i :$Q \times \Sigma \to 2^Q$, δ_r:$Q \times Q \times \Sigma \to 2^Q$
  - Can be determinized: blow-up 2^{n2}

- Appealing theoretical properties
  - Effectively closed under various operations (union, intersection, complement, concatenation, prefix-closure, projection, Kleene-* …)
  - Decidable decision problems: membership, language inclusion, language equivalence …
  - Alternate characterization: MSO, syntactic congruences

Linguistic Annotated Data



Linguistic data stored as annotated sentences (eg. Penn Treebank)

Sample query: Find nouns that follow a verb which is a child of a verb phrase

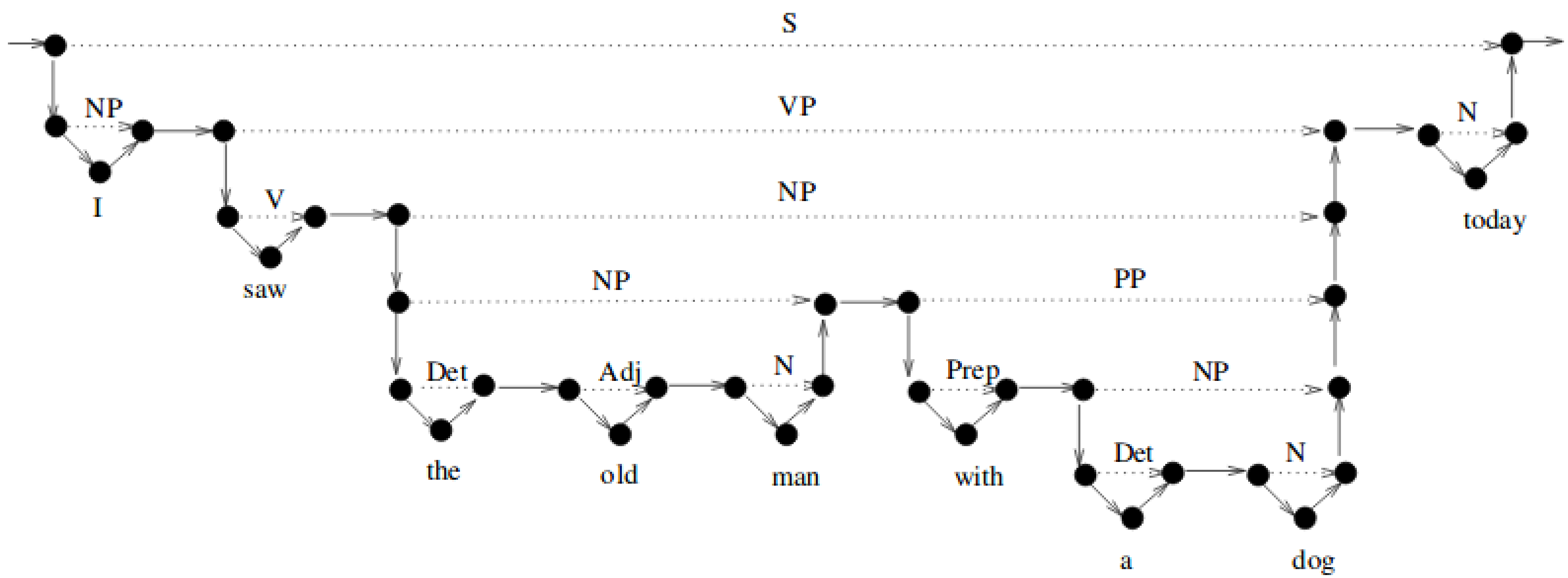


Fig. 4. Parsed sentence as a nested word

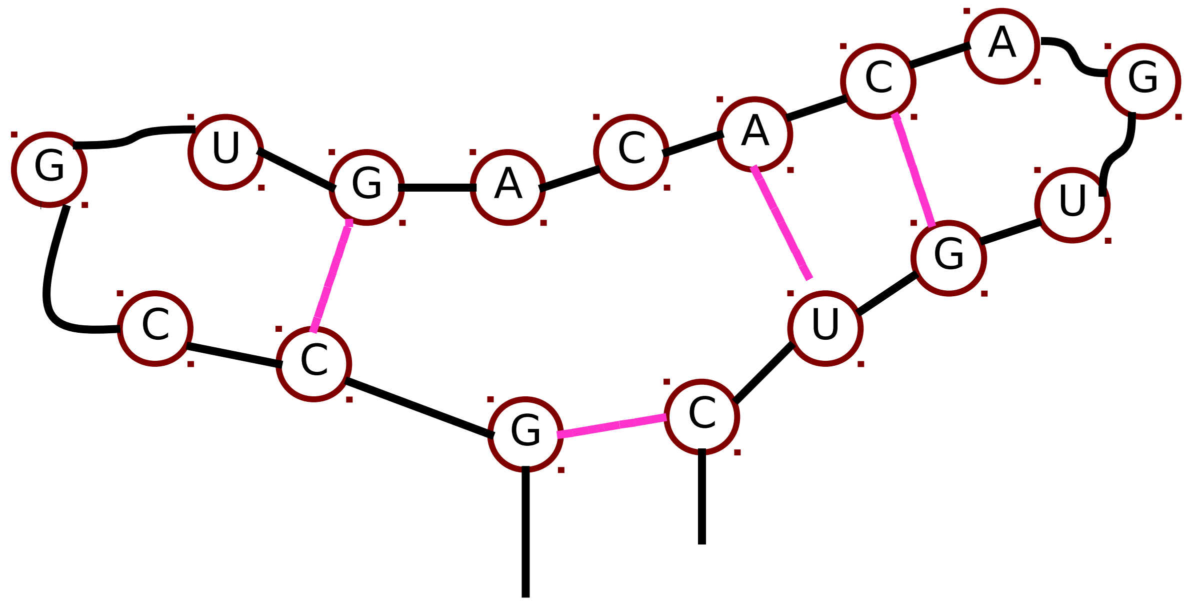
```
<S>
  <NP> I
  </NP>
  <VP>
    <V> saw
    </V>
    <NP>
      <NP>
         <Det> the
         </Det>
         <Adj> old
         </Adj>
         <N> man
         </N>
      </NP>
      <PP>
         <Prep> with
         </Prep>
         <NP>
            <Det>  a
            </Det>
            <N>  dog
            </N>
         </NP>
      </PP>
    </NP>
  </VP>
  <N>  today
  </N>
</S>
```

Fig. 5. XML representation of parsed sentence

RNA as a Nested Word

Primary structure: Linear sequence of nucleotides (A, C, G, U)

Secondary structure: Hydrogen bonds between complementary nucleotides (A-U, G-C, G-U)



In literature, this is modeled as trees.

Algorithmic question: Find similarity between RNAs using edit distances

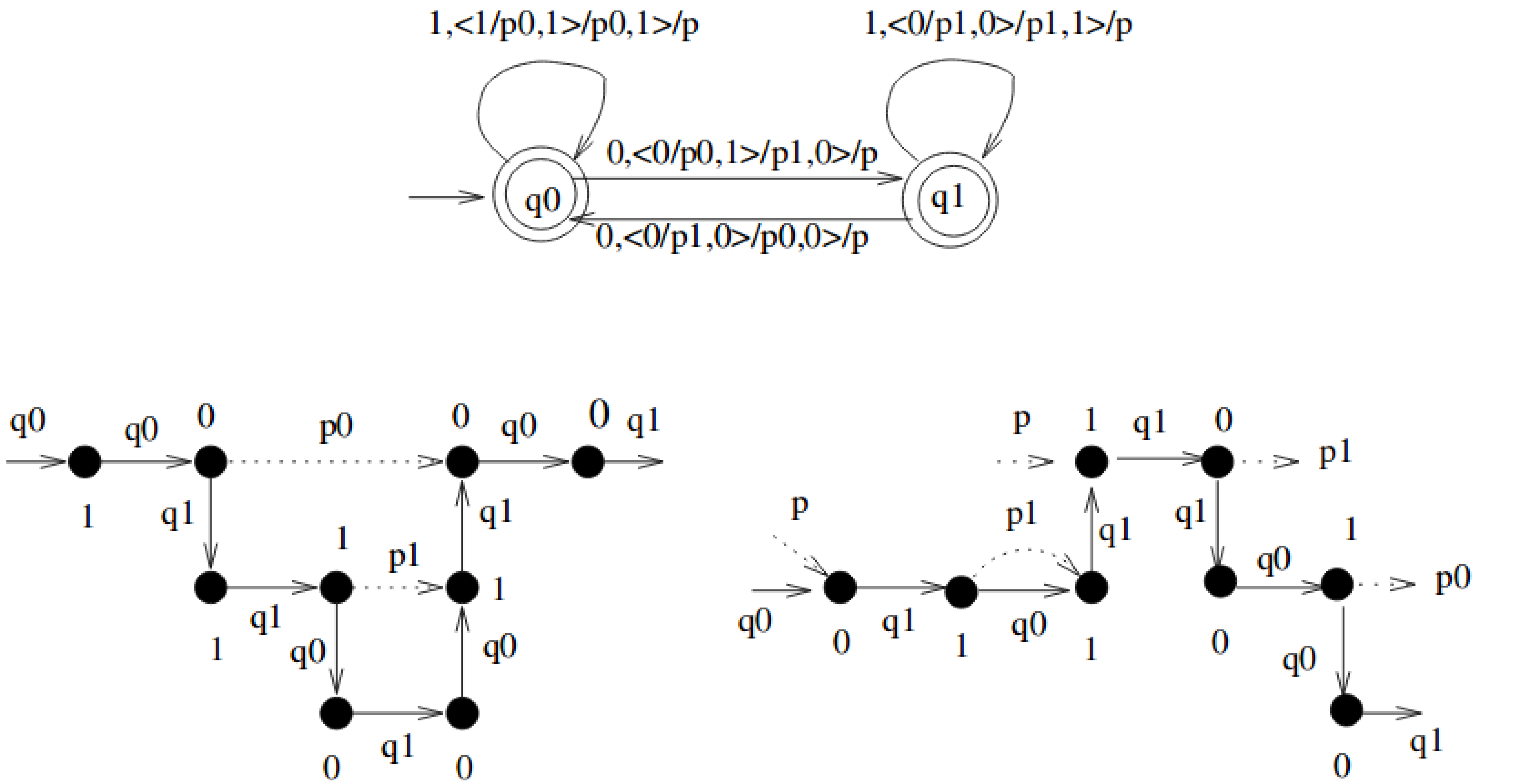


Fig. 6. Example of an NWA and its runs

References

- Adding Nested Structure to Words, Rajeev Alur and P. Madhusudan, JACM 2009
- Adding Nested Structure to Words, Rajeev Alur and P. Madhusudan, DLT 2006
- Marrying Words and Trees (slides), Rajeev Alur
- Nested Word Automata (slides), Christian Schilling