

On the Practice of B-ing Earley

Dan Zingaro

zingard@mcmaster.ca

McMaster University

August 2007

What's With the Title?

- Earley: algorithm for testing whether a sentence can be recognized via an arbitrary context-free grammar in worst-case $O(n^3)$
- Other algorithms require special normal form (CYK, 1965), have more complicated structures (Unger, 1968), or parse only some grammars (LL , LR)
- Applications: probabilistic language processing (Stolcke, 1995), compiler-compilers (Accent)
- Other formalisms: Earley's thesis (1968, informal), Sikkel (1998, parsing schemata), Jones (1972, origins of refinement)

Current Interest in Earley

- Previously thought Earley was too slow compared to *LL* and *LR*
- Aycok and Horspool (2002) showed *LR* is only 1.9x faster than Earley
- Claim: time difference is not noticeable, might as well support full power of CFGs

What's With the Title? ...

- B: formal method for specifying and refining software
- Expressive: sets, sequences, relations, functions
- Want to prove properties of algorithm but also derive it from specification
- Modularization: separate development and understanding of components
- Typecheckers, provers, model checkers available

What is a Recognizer?

- Derivable: \Rightarrow^*
- Recognizer returns true if $Root \Rightarrow^* \text{sentence}$

Nonterminals == set of nonterminal ;

Terminals == set of terminal ;

Symbols == *Terminals* \cup *Nonterminals* ;

Root \in *Nonterminals* ;

productions \in *Nonterminals* \leftrightarrow seq(*Symbols*)

Derivability

MACHINE *recm*

SEES *gram, sent*

DEFINITIONS

directlyDerivable ==

{*x, y* |

($\exists \mu, \sigma, \nu, \tau$).

($x = (\mu \frown [\sigma] \frown \nu) \wedge (y = \mu \frown \tau \frown \nu) \wedge (\sigma \mapsto \tau \in \text{productions})$);

derivable == *closure*(*directlyDerivable*)

OPERATIONS

ans \longleftarrow *isSentence* =

ans := bool ([*Root*] \mapsto *sentence* \in *derivable*)

Earley Ingredients

- Length of sentence: n
- Create $n + 1$ state sets; one per sentence position + initial set
- State: $[A \rightarrow \alpha.\beta, j]$

$state == Nonterminals \times seq(Symbols) \times seq(Symbols) \times \mathbb{N};$

$s \in (0 .. size(sentence)) \rightarrow \mathbb{P}(state)$

-
-
-

Earley Execution

Grammar Rules: $S \rightarrow A, A \rightarrow a, A \rightarrow b$

Sentence: a

$s(0)$:



Earley Execution

Grammar Rules: $S \rightarrow A$, $A \rightarrow a$, $A \rightarrow b$

Sentence: a

$s(0)$:

- 1. $[S \rightarrow \bullet A, 0]$ (basis)



Earley Execution

Grammar Rules: $S \rightarrow A$, $A \rightarrow a$, $A \rightarrow b$

Sentence: a

$s(0)$:

- 1. $[S \rightarrow \bullet A, 0]$ (basis)
- 2. $[A \rightarrow \bullet a, 0]$ (predict from 1)
- 3. $[A \rightarrow \bullet b, 0]$ (predict from 1)

$s(1)$:

Earley Execution

Grammar Rules: $S \rightarrow A$, $A \rightarrow a$, $A \rightarrow b$

Sentence: a

$s(0)$:

- 1. $[S \rightarrow \bullet A, 0]$ (basis)
- 2. $[A \rightarrow \bullet a, 0]$ (predict from 1)
- 3. $[A \rightarrow \bullet b, 0]$ (predict from 1)

$s(1)$:

- 4. $[A \rightarrow a\bullet, 0]$ (scan from 2)

Earley Execution

Grammar Rules: $S \rightarrow A$, $A \rightarrow a$, $A \rightarrow b$

Sentence: a

$s(0)$:

- 1. $[S \rightarrow \bullet A, 0]$ (basis)
- 2. $[A \rightarrow \bullet a, 0]$ (predict from 1)
- 3. $[A \rightarrow \bullet b, 0]$ (predict from 1)

$s(1)$:

- 4. $[A \rightarrow a\bullet, 0]$ (scan from 2)
- 5. $[S \rightarrow A\bullet, 0]$ (complete from 4)

Key Invariants

- I recognizes $sentence(f+1 .. ind)$

$$\begin{aligned} & (\forall a1, l1, r1, f1, ind) . (ind < ii \wedge (a1 \mapsto l1 \mapsto r1 \mapsto f1) \in state \wedge \\ & (a1 \mapsto l1 \mapsto r1 \mapsto f1) \in s(ind) \implies \\ & (l1 \mapsto sentence(f+1 .. ind)) \in derivable) \wedge \end{aligned}$$

- If next symbol recognizes $sentence(j+1 .. z)$, item with dot moved is in state set z

$$\begin{aligned} & (\forall z, j, a1, l1, r1, f1) . (z < ii \wedge \\ & j \leq z \wedge (a1 \mapsto l1 \mapsto r1 \mapsto f1) \in state \wedge (a1 \mapsto l1 \mapsto r1 \mapsto f1) \in s(j) \wedge \\ & ([first (r1)] \mapsto sentence(j+1 .. z)) \in derivable \implies \\ & ((a1 \mapsto (l1 \frown [first (r1)])) \mapsto tail (r1) \mapsto f1)) \in s(z)) \end{aligned}$$



Refining to Lists

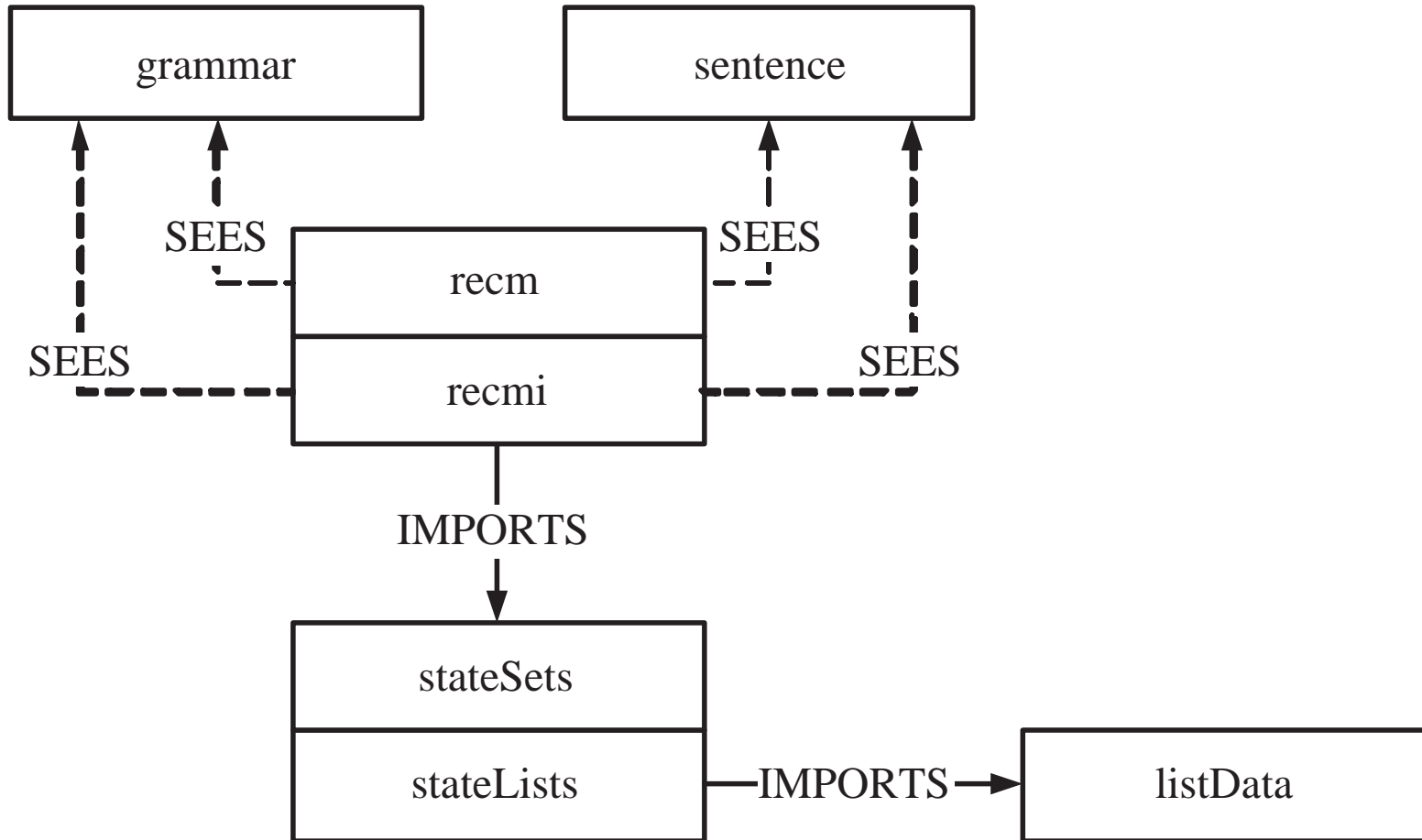
- Specification of algorithm uses transitive closure of Earley operations to add all states
- Implementation: use something more concrete
- Idea: move linearly through the list; apply Earley operation to items as we go
- Works when there are no ϵ -productions, but we run into trouble when there are
- Problem: completer misses states



Refining to Lists...

- Keep traversing list over and over until nothing new (inefficient)
- Keep track of all previous completers; run them again after processing (yuck, dynamic data structure)
- Aycok, Horspool 2002: traverse once, modify predictor to ensure no item is missed
- Linking invariant: states and sets are equal

Structure of Development



Conclusion

- First formal verification of context-free recognizer
- Tools: ProB (for model checking), B4free (typechecking, automatic proving), hand-proof of correctness
- Invariants also lead to discovery of an optimization of the list refinement,
- Development synthesizes Earley and Ayrcock-Horspool into a single framework

Future Work

- Formalize parser version of recognizer
 - ◇ Purpose: investigate when and why Earley produces incorrect parse trees (Tomita, 1986)
- Develop a parser based on Parsing Expression Grammars
 - ◇ Purpose: inherently unambiguous (does this simplify things?), characterize the languages they can deal with
- CNF-based algorithms like CYK: do restrictions lead to more direct formal proofs?
- Any (simple) questions?

Epsilon Example

Grammar rules: $S \rightarrow AA, A \rightarrow \epsilon$

$S \rightarrow \bullet AA$, 0 Basis

$A \rightarrow \epsilon$, 0 Predictor

$S \rightarrow A \bullet A$, 0 Completer

$S \rightarrow AA \bullet$, 0 **MISSED!**

-
-
-

Non-LR Java Fragment

FieldDecl \rightarrow [FieldMods] Type Vars

FieldMods \rightarrow FieldMod | FieldMods FieldMod

MethodHeader \rightarrow [MethodMods] ResultType MethodDeclarator

Try parsing `public static int...`