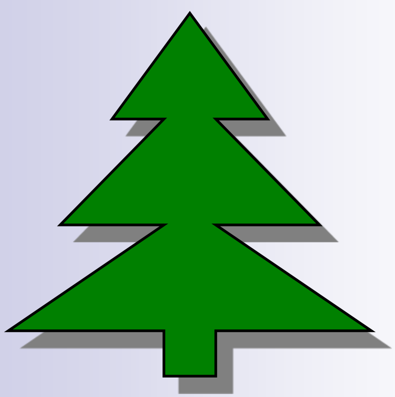
The Complexity of Tree Transducer Output Languages

FSTTCS 2008, Bengaluru



The Univ. of Tokyo Kazuhiro Inaba

NICTA, and UNSW Sebastian Maneth

“Complexity of Output Languages”

- Given…
  - A language
    $L \subseteq T_\Sigma$ (Trees over Σ)
  - A relation (nondeterministic translation)
    $\tau \subseteq T_\Sigma \times T_\Delta$ (from T_Σ to T_Δ)

- What is the complexity of the language
  $\tau(L) \subseteq T_\Delta$?

(i.e., for $t \in T_\Delta$, how is it computationally hard
to determine whether $t \in \tau(L)$ or not?)

Classic Results

- τ: Program of Turing-Machine
  - Undecidable

- L : Regular String Language
- τ: Nondeterministic Finite State Transduction
  - τ(L) is regular!
    - → The membership of τ(L) is solved in O(n) time, O(1) space

  - Corollary: for τ∈ <u>Finitely Many Compositions</u> of Nondeterministic FST, τ(L) is regular

Trees?

- L : Regular Tree Language
- τ: Finitely Compositions of
  Nondet. Finite-State Tree Transducers
  - Beyond Regular Tree Language
    - (Intuitively…) Due to Copying
      - $\tau(t) \rightarrow x(t, t)$ is an instance of FSTT
  - In DSPACE(n) [Baker1978]
    - i.e., Deterministic Context-Sensitive

Recent Result [Maneth2002, FSTTCS]

- L : Regular Tree Language
- τ: Finite Compositions of
  Total Deterministic Macro Tree Transducers
  - == Tree Transducers extended with "accumulating parameters" for each state
  - In DSPACE(n)
    - Still, Deterministic Context-Sensitive

Today's Target!

- L : Regular Tree Language
- τ: Finite Compositions of
  Nondeterministic Macro Tree Transducer
  - Is it still context-senstive? – Yes. NSPACE(n)
  - What about the time complexity? – NP-complete

Outline

- What is/Why Macro Tree Transducers?
- Review of the Proof for Deterministic Case
- “Garbage-free” Lemma
- “Translation Membership” Problem
- Summary

Macro Tree Transducer (MTT)

- Q : Finite Set of States
- q0: Initial State
- Σ : Input Alphabet
- Δ : Output Alphabet
- R : Set of Rewrite Rules of form:

  $\langle q, \sigma(x_1,\ldots,x_k)\rangle(y_1,\ldots, y_m) \rightarrow r$

  where $r ::= \delta(r, \ldots, r) \mid \langle q, x_i\rangle(r, \ldots, r) \mid y_j$

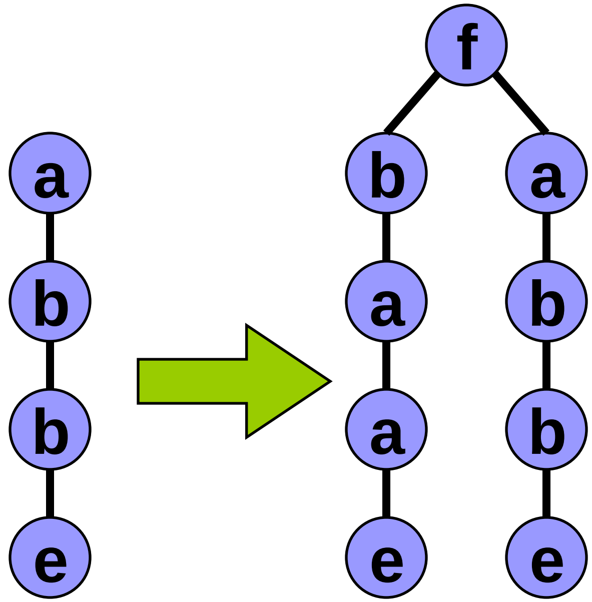
Example of an MTT

- <q0, a(x)>() → f(<q1, x>(a(e)), <q2,x>())
- <q0, b(x)>() → f(<q1, x>(b(e)), <q2,x>())

- <q1, a(x)>(y) → <q1, x>(a(y)))
- <q1, b(x)>(y) → <q1, x>(b(y)))
- <q1, e>(y) → y

- <q2, a(x)>() → a(<q2,x>())
- <q2, b(x)>() → b(<q2,x>())
- <q2, e>() → e

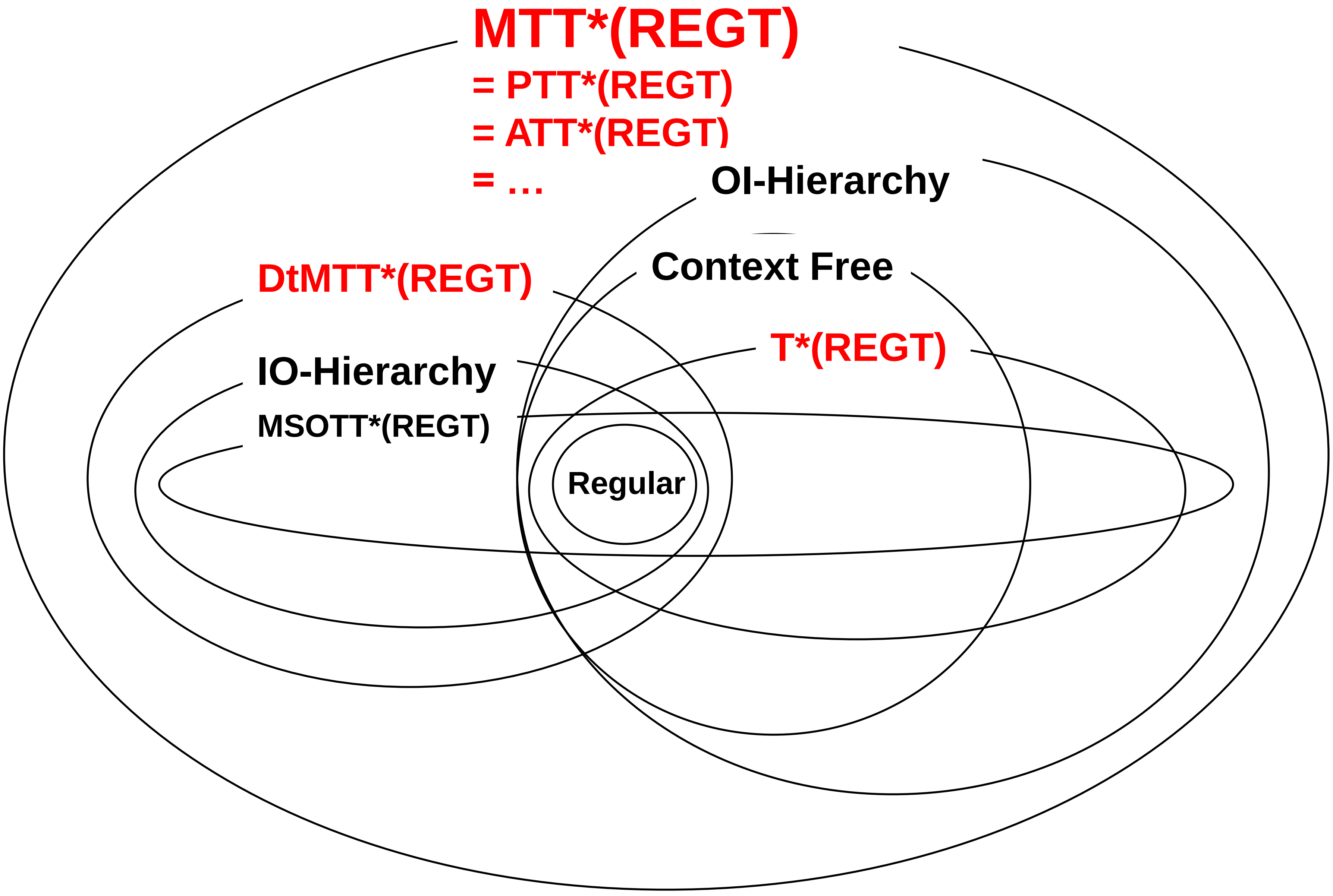
<q0, a(b(b(e))>()
→ f(<q1,b(b(e))>(a(e)), <q2,a(e)>())
→ f(<q1,b(e)>(a(a(e)), <q2,a(e)>())
→ f(<q1,e>(a(a(a(e))), <q2,a(e)>()) → …



(Choice of Semantics)

- Functional Programming + Laziness + Nondeterminism ☺

- We take the Runtime-Choice Semantics:
  - <coin, a> → 0 | 1
  - <twocoins, a>(y) → c(y, y)
  - <twocoins, a>(<coin,a>()) →*
    { c(0,0), c(0,1), c(1,0), c(1,1) }
- Because of its composability: MTT ; LT ⊆ MTT


MTT*(REGT)
= PTT*(REGT)
= ATT*(REGT)
= …
OI-Hierarchy
Context Free
DtMTT*(REGT)
T*(REGT)
IO-Hierarchy
MSOTT*(REGT)
Regular

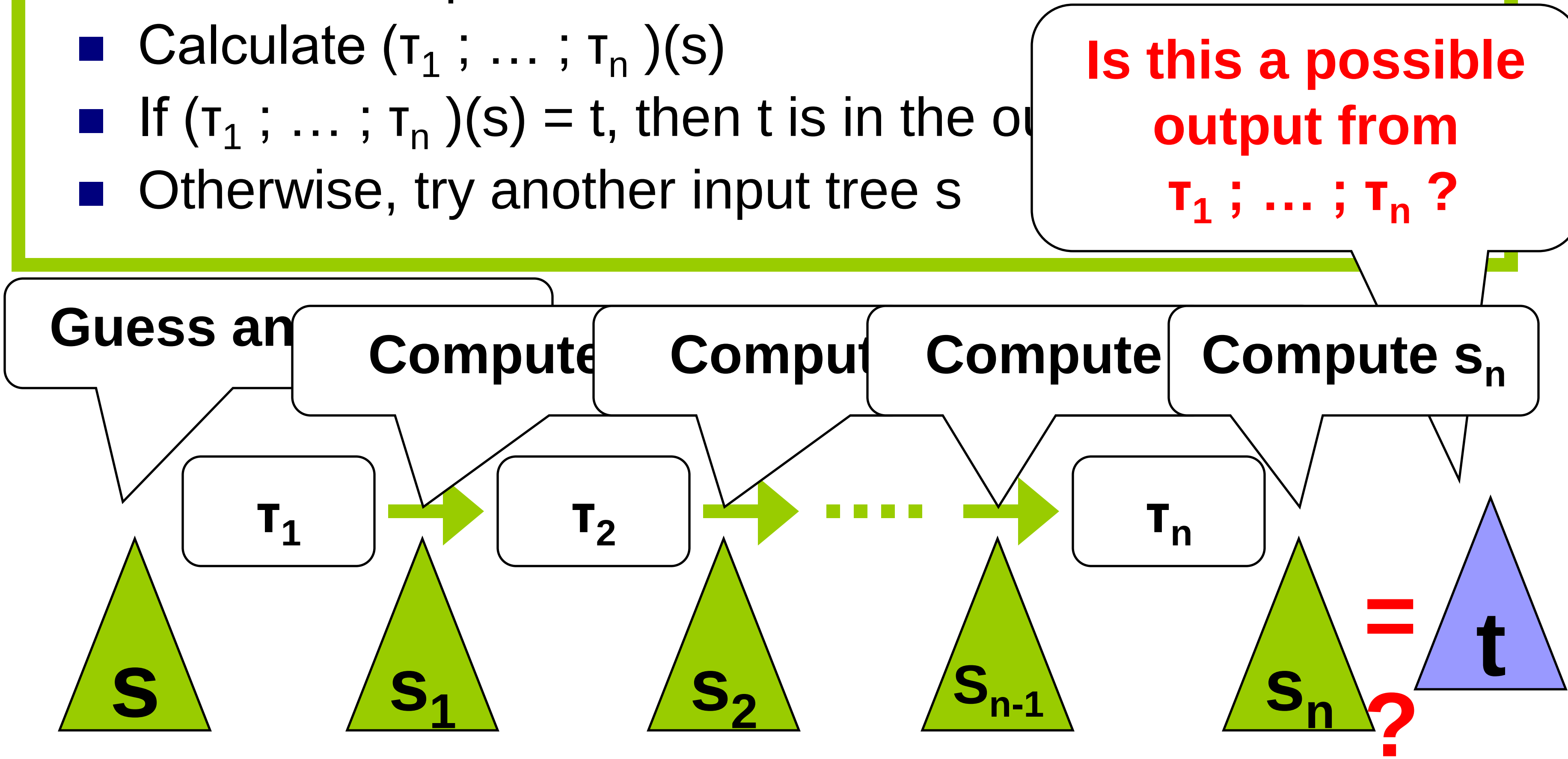
Review:
DSPACE(n) Membership for Det. MTTs

- Given a (fixed) pair of
  - Input regular language L and
  - Composition sequence τ_1 ; … ; τ_n of total deterministic mtts
- and a tree t,
- How can we test $t \in (\tau_1 ; \ldots ; \tau_n)(L)$ in linear space wrt |t|?

Review:

DSPACE(n) Membership for Det. MTTs

- Guess the input $s \in L$
- Calculate $(\tau_1 ; \ldots ; \tau_n)(s)$
- If $(\tau_1 ; \ldots ; \tau_n)(s) = t$, then t is in the ou
- Otherwise, try another input tree s

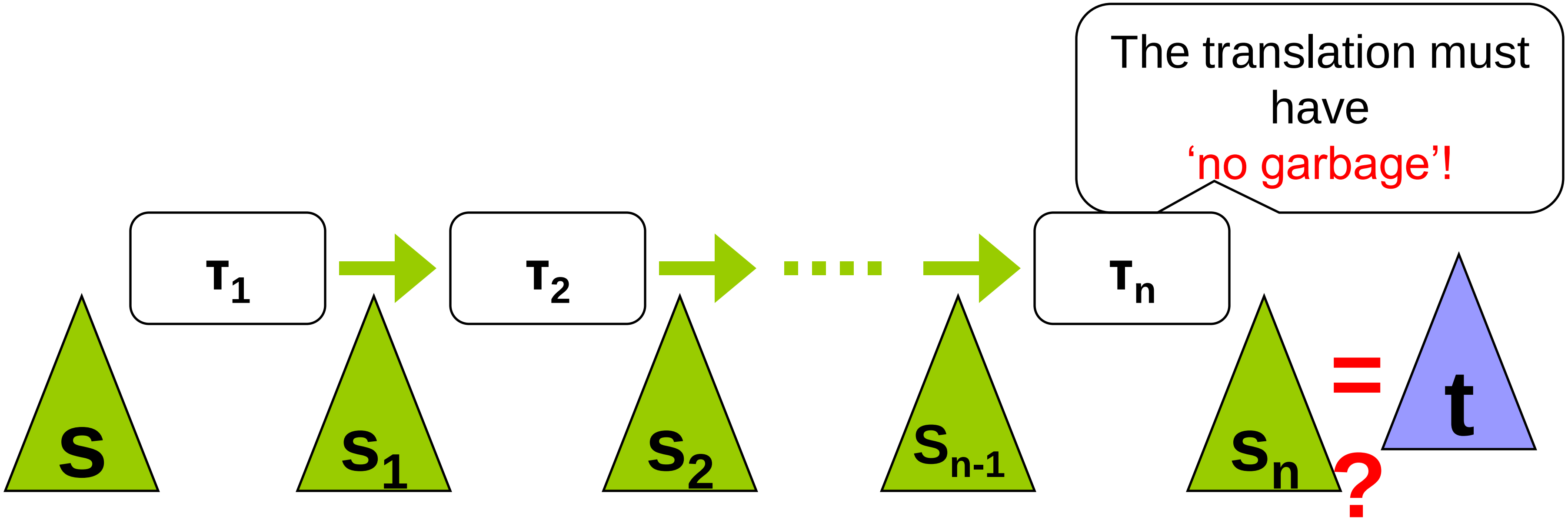


Review:

DSPACE(n) Membership for Det. MTTs

- In order to carry out the algorithm in DSPACE(|t|) …
  - The sizes $|s|$, $|s_1|$, $|s_2|$, …, $|s_n|$ must be linearly bounded by $|t|$
    - i.e., there must be a constant c independent from t s.t. $|s| \leqq c|t|$

  - Each step τ of the computation must be done in linear space



Review:

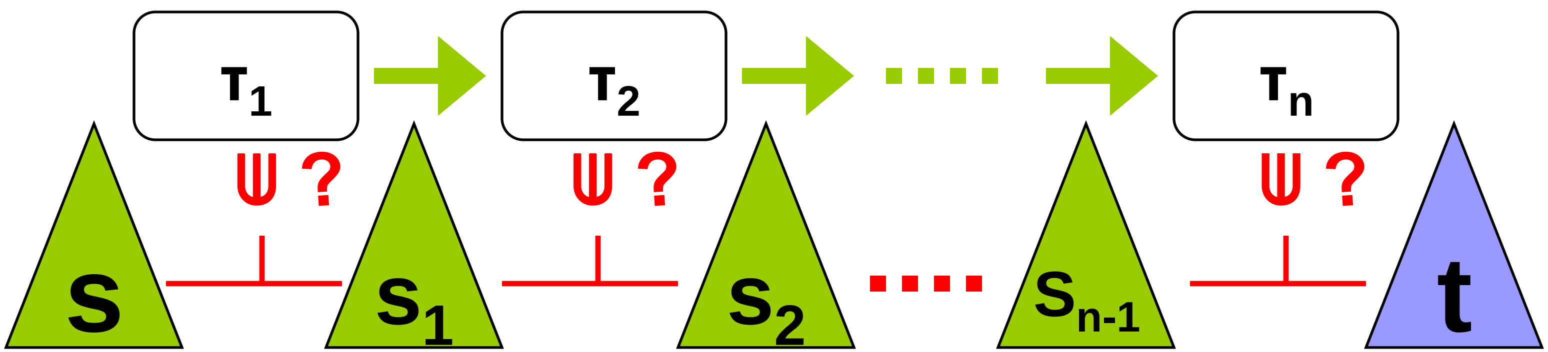
DSPACE(n) Membership for Det. MTTs

- ‘Garbage-Free’ Lemma
  - For any input language L and mtts $\tau 1, \ldots, \tau n$, there exists L’ and $\tau' 1, \ldots, \tau' n$ such that
    $(\tau_1;\ldots;\tau_n)(L) == (\tau'_1;\ldots;\tau'_n)(L')$
    and every τ'_i is ‘non-deleting’ ($|in| \leqq 2|out|$)

- Linear Time (and Space) Computation
  - For any total deterministic mtt τ and a tree s, $\tau(s)$ can be computed in time $O(\ |s| + |\tau(s)|\ )$ (already known as a folklore result)

NSPACE(n)/NP Output Membership for Nondeterministic MTTs

- Guess the input $s \in L$
  and all the intermediate trees $s_1, \ldots, s_{n-1}$
- Check whether
  $(s,s_1) \in \tau_1, (s_1,s_2) \in \tau_2, \ldots, (s_{n-1}, t) \in \tau_n$
- If it is, then t is in the output language!
- Otherwise, try another $s, s_2, \ldots, s_{n-1}$



Key Lemmas

- ‘Garbage-Free’ Lemma—Nondet. Version

- NP/NSPACE(n) “Translation Membership” for a single mtt translation

Key Lemma (1):
‘Garbage-Free’ Lemma—Nondet. Version

- Basic Idea
  - “Factor out” the deletion
  - $\tau_1 \;;\; \tau_2 \;==\; \tau_1 \;;\; (D \;; \tau'_2)$

    $== (\tau_1 \;;\; D) \;;\; \tau'_2$

    $== \rho_1 ; \tau'_2$

Decompose τ_2
to ‘deleting part’ D
and ‘nondeleting’ τ'_2

Associativity

Compose τ_1 with D

Three Types of Deletion

Lemma:
If no erasing, input-deleting, or skipping rule is used during the computation, then $|in| \leqq 2|out|$

- "Erasure"
  - $\langle q,\sigma\rangle(y_1, y_2) \rightarrow y_1$
  - No new output node is generated at this σ node. Only returning its parameter.
- "Input-Deletion"
  - $\langle q, \sigma(x_1, x_2)\rangle() \rightarrow \delta(\langle q, x_1\rangle())$
  - Discarding the "x_2" subtree!
- "Skipping"
  - $\langle q, \sigma(x_1)\rangle() \rightarrow \langle q, x_1\rangle()$
  - Occurs only at monadic node. No new output is generated here. Just going down to its child node.

Eliminating The Three Types of Deletion

- Achieved by heavily manipulating the rules
  - For details, please consult the paper

- One of the difficulties compared to the deterministic case: Inline-Expansion
  - $\langle q, a\rangle(y) \rightarrow y$
  - $\langle q, b(x_1,x_2)\rangle \rightarrow c(\ \langle p,x_1\rangle(\langle q,x_2\rangle(e))\ )$

    (Assume we know that ‘b’’s child is always ‘a’)

  - $\langle q, b(x_1,x_2)\rangle \rightarrow c(\ \langle p,x_1\rangle(\ e\ )\ )$

With Nondeterminism, Inline-Expansion is Not Easy

- <q, a>() → e
- <q, a>() → f
- <q, b(x)>() → <p,x>(<q,x>())
- <p, a>(y) → c(y, y)

<q, b(a)>()
→ <p,a>(<q,a>())
→ c(<q,a>(), <q,a>())
→ c(e, f)

Different

<q, b(a)>()
→ <p,a>(e) → c(e, e)
or
→ <p,a>(f) → c(f, f)

- <q, a>() → e
- <q, a>() → f
- <q, b(x)>() → <p,x>(e)
- <q, b(x)>() → <p,x>(f)
- <p, a>(y) → c(y, y)

Solution: “MTT with Choice and Failure”

- We have extended MTTs with “inline” nondeterminism
  - Allows inline-expansion for free!
  - Actually, we prove the output language complexity for mtt-cfs

<q, b(a)>()
→ <p,a>(+(e,f))
→ c(+(e,f), +(e,f))
→ c(e, f)

- <q, a>() → e
- <q, a>() → f
- <q, b(x)>() → <p,x>(+(e, f))
- <p, a>(y) → c(y, y)

Key Lemma (2): “Translation Membership” of single τ_i

- Given a pair (s_{i-1}, s_i) of trees,
  we can determine whether $(s_{i-1}, s_i) \in \tau_i$
  in linear-space & polynomial time wrt $|s_{i-1}|+|s_i|$
  in nondet. Turing machine

- Naively Applying the folklore deterministic computation takes $O(|s_{i-1}| + |\tau(s_{i-1})|)$ time/space
  → New Idea is Necessary

“Translation Membership” of single τ_i

- Naively Applyi…
  for D…

OK. Similar decomposition works also for Nondet. MTTs

Bad. Nondet. Linear MTTs may read each node multiple times

Need More Sophisticated Compression!

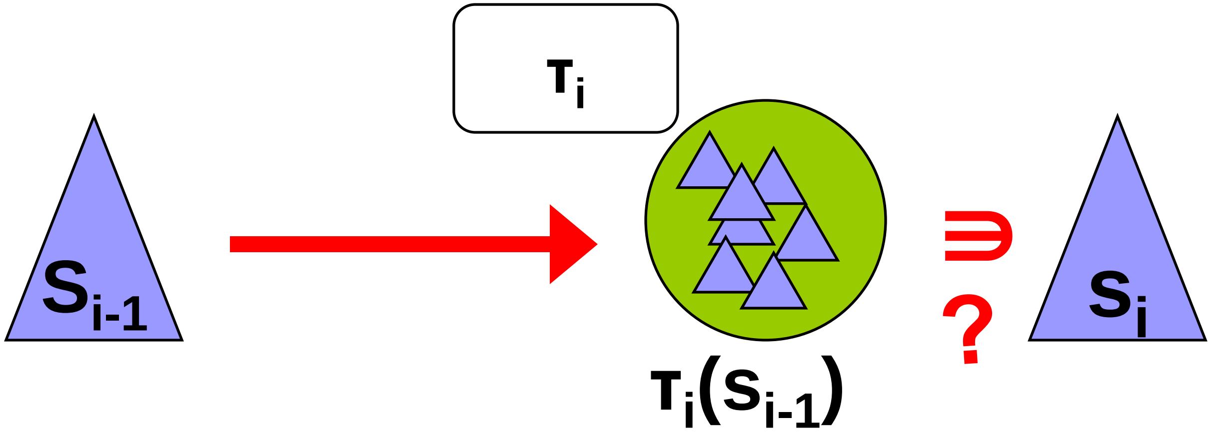
  - …ecomp…ion of an MTT into Linear MTTs (…input varia…e x_i occurs at most once in each rule),
  - …and the fa…t that deterministic linear MTTs read each input no…e at most once,
  - …which allows to compress the output tree as a DAG for both saving space and sharing computations

Example: Linear Nondet. MTT Reading Some Node Twice

- <q, b(x1,x2)>() → <p,x1>(<q,x2>())
- <q, a>() → e
- <q, a>() → f
- <p, a>(y) → g(y, y)

Solution: Compression by Context-Free Tree Grammar

- The set all outputs $\tau_i(s_{i-1})$ of an MTT can be represented by a CFTG of size proportional to $|s_{i-1}|$ [MB04]



- Navigation (**up**,1stchild, nextsibl) on the compressed representation is efficient for linear mtts

Summary

- Composition sequence τ_1 ; …; τ_n of mtts can be converted to an equivalent ‘garbage-free’ composition
- Translation Membership of any mtt is in NP/NSPACE(n)

- → Altogether, the output language complexity of mtt-compositions is NP/NSPACE(n)
  - Corollary: OI-hierarchy, PTT*(REGT), ATT*(REGT), … is in NP/NSPACE(n)
- Current Status (Unpublished): NSPACE(n)→DSPACE(n)