Introduction to Tree Language Theory

Hitoshi Ohsaki

National Institute of Advanced Industrial Science and Technology (AIST)
Acknowledgment. The author would like to thank prof. Hiroyuki Seki at NAIST in Japan for his advices and detailed comments overall the lecture materials. I’m also grateful to many comments from people whom I met at University of Illinois at Urbana-Champaign, in particular, Peter Csaba Olveczky at University of Oslo, prof. José Meseguer and his students including Joe Hendrix, Mike Katelman, Camilo Rocha, Ralf Sasse. The first draft of this text has been accomplished when I visited at UIUC for a year since June 2007.
I. Words
Alphabet

A finite set $\Sigma = \{ s_0, \ldots, s_n \}$ of symbols (i.e. concrete and visibly or otherwise recognizable representations) is called an alphabet if every non-empty finite sequence over $\Sigma$ is uniquely decomposed to elements in $\Sigma$.

Elements in the alphabet $\Sigma$ are called characters or letters.

Example

Consider $\Sigma_1 = \{ 1, 11 \}$ $\Sigma_2 = \{ 01, 11 \}$ $\Sigma_3 = \{ 0, 01, 10 \}$

$\Sigma_1$ is not an alphabet since 11 is formed by either 11 or 1 (and) 1

$\Sigma_2$ is an alphabet

$\Sigma_3$ is not an alphabet, e.g., since 010 is decomposed in two ways 0 1 0 and 01 0

Binary & unary encoding

Consider the alphabet $\Sigma_1 = \{ a, b, c \}$

$$\begin{aligned}
&\text{a b c} \\
&\psi_1 \downarrow \quad \text{let } \psi_1 : \text{mapping of } a \mapsto 1, \ b \mapsto 10, \ c \mapsto 100 \\
&110100 \quad \text{(binary encoding over } \Sigma_2 = \{ 0, 1 \}\text{)} \\
&\psi_2 \downarrow \quad \text{let } \psi_2 : \text{mapping of } n \mapsto 1^{\text{decimal}}(n) \quad \text{(sequence of 1's)} \\
&1^{\text{decimal}}(110100) \quad \text{(unary encoding over } \Sigma_3 = \{ 1 \}\text{)}
\end{aligned}$$

Remark

$\psi_1 : \psi_1^{-1}(\psi_1(v)) = v$ $\&$ $\psi_1(\psi_1^{-1}(w)) = w$ if $w \in \{ 1, 10, 100 \}$

$\psi_2 : \psi_2^{-1}(\psi_2(v)) = v$ $\&$ $\psi_2(\psi_2^{-1}(w)) = w$
Finite automata

finite automaton \((\Sigma, Q, q_0, Q_{\text{fin}}, \Delta)\)

\[
\begin{align*}
\Sigma & : \text{alphabet} \\
Q & : \text{finite set of state symbols} \\
q_0 & : \text{initial state such that } q_0 \in Q \\
Q_{\text{fin}} & : \text{final states such that } Q_{\text{fin}} \subseteq Q \\
\Delta & : \text{finite set of transition rules with the following forms}
\end{align*}
\]

\[
p \xrightarrow{a} q \quad p \to q \quad (p, q \in Q, a \in \Sigma)
\]

**Note**

\[
p \xrightarrow{a} q \text{ means “state } p \text{ moves to } q \text{ reading character } a”
\]

\[
p \to q \text{ means “state } p \text{ moves to } q”
\]

Finite automata (as machines)

Given a tape \(a_0 a_1 \cdots a_i a_{i+1} \cdots a_n\) and finite automaton \(\mathcal{A}\)

\[
\begin{align*}
\begin{array}{c}
q_0 \\
a_0 a_1 \cdots a_i a_{i+1} \cdots a_n
\end{array} & \xrightarrow{a_0} \begin{array}{c}
q_1 \\
a_0 a_1 \cdots a_i a_{i+1} \cdots a_n
\end{array} \\
\begin{array}{c}
q_i \\
a_0 a_1 \cdots a_i a_{i+1} \cdots a_n
\end{array} & \xrightarrow{a_i} \begin{array}{c}
q_{i+1} \\
a_0 a_1 \cdots a_i a_{i+1} \cdots a_n
\end{array} \\
\begin{array}{c}
q_{n+1}
\end{array}
\end{align*}
\]

1st step: \(q_0 \xrightarrow{a_0} q_1\) such that \(q_0\) is initial state

\(i\)-th step: \(q_i \xrightarrow{a_i} q_{i+1}\)

If the above move ends in \(a_0 a_1 \cdots a_i a_{i+1} \cdots a_n\) and \(q_{n+1}\) is a final state, we say “the input tape is accepted by \(\mathcal{A}\)”
Words and Languages

Given an alphabet $\Sigma$

word over $\Sigma$ ⟩ ⟩ a finite sequence of characters from $\Sigma$
empty word $\varepsilon$ ⟩ ⟩ empty sequence of character

language over $\Sigma$ ⟩ ⟩ a subset of words over $\Sigma$
$\Sigma^*$ ⟩ ⟩ a set of all words over $\Sigma$
$\Sigma^+$ ⟩ ⟩ a set of all non-empty words, $\Sigma^* - \{\varepsilon\}$

Given a finite automaton $A$ with $\Sigma$

word accepted by $A$ ⟩ ⟩ word, as input tape, accepted by $A$
language accepted by $A$ ⟩ ⟩ a set of words accepted by $A$

For instance, we say “a language $L$ over $\Sigma$ is accepted by $A$,“ denoted $\mathcal{L}(A)$, if $L = \{ w \in \Sigma^* | w$ is accepted by $A \}$

Example

Define transition rules

$\Delta_1 : q_0 \xrightarrow{a} q_0 \quad q_0 \xrightarrow{b} q_0 \quad q_0 \xrightarrow{c} q_0$
$\Delta_2 : \emptyset$
$\Delta_3 : q_0 \xrightarrow{a} q_0 \quad q_1 \xrightarrow{b} q_1 \quad q_2 \xrightarrow{c} q_2$
\hspace{1cm} $q_0 \xrightarrow{\quad} q_1 \quad q_1 \xrightarrow{\quad} q_2$

for the finite automata with the alphabet $\Sigma = \{a, b, c\}$. Let

$A_1 = (\Sigma, \{q_0\}, q_0, \{q_0\}, \Delta_1)$
$A_2 = (\Sigma, \{q_0\}, q_0, \{q_0\}, \Delta_2)$
$A_3 = (\Sigma, \{q_0, q_1, q_2\}, q_0, \{q_2\}, \Delta_3)$

then

$\mathcal{L}(A_1) = \Sigma^* \quad \mathcal{L}(A_2) = \{\varepsilon\} \quad \mathcal{L}(A_3) = \{a^\ell b^m c^n | \ell, m, n \geq 0\}$
Deterministic finite automata (DFA)

finite automaton \((\Sigma, Q, q_0, Q_{\text{fin}}, \Delta)\) is deterministic if

1. \(\Delta\) contains transition rules with the following form only:
   \[ p \xrightarrow{a} q \]

2. there are no transition rules \(p \xrightarrow{a} q_1\) and \(p \xrightarrow{a} q_2\) with \(q_1 \not\equiv q_2\)

**Proposition**

Given a finite automaton \(A\), one can construct a DFA \(B\) such that \(L(A) = L(B)\) i.e., DFA \(B\) accepts the language accepted by FA \(A\)

**Proof**

Let \(A = (\Sigma, Q, q_0, Q_{\text{fin}}, \Delta)\), the procedure for constructing DFA consists of two steps:

1. eliminating rules of the form \(p \rightarrow q\) from \(\Delta\)
2. eliminating non-determinism in rules of \(p \xrightarrow{a} q_1\) and \(p \xrightarrow{a} q_2\) with \(q_1 \not\equiv q_2\)

(proof cont’d)

(1) Repeat the following computation until no more rules can be added to \(\Delta\):

   if there is a pair of transition rules in \(\Delta\) either \(p \xrightarrow{a} q\), \(q \rightarrow r\) or
   \[ p \rightarrow q, \; q \xrightarrow{a} r, \]
   then add \(p \xrightarrow{a} r\) to \(\Delta\)

**Note 1**

The above computation terminates because the number of transition rules of the form \(p \xrightarrow{a} q\) in \(\Delta\) must be \(|Q|^2 \times |\Sigma|\) or less, and this is the upper limit on the number of the loops.

(2) Take \(2^Q\), which is the power set of \(Q\), and let \(Q_{\text{dfin}} = \{ S \in 2^Q \mid S \cap Q_{\text{fin}} \neq \emptyset \}\). If \(A\) accepts \(\varepsilon\), add \(\{q_0\}\) to \(Q_{\text{dfin}}\). Define finite automaton \(B = (\Sigma, 2^Q, \{q_0\}, Q_{\text{dfin}}, \Delta_d)\) based on \(A\) with \(\Delta\) obtained in (1), where for each two states \(S,T\) in \(2^Q\) and character \(a\) in \(\Sigma\),

   \[ S \xrightarrow{a} T \text{ in } \Delta_d \]

if \(T = \{ q \mid p \in S \text{ and } p \xrightarrow{a} q \in \Delta \}\) and \(T \neq \emptyset\).

**Note 2**

In the above construction, the reverse \((p_i \rightarrow p_{i+1}, \; p_m \xrightarrow{a} q_1, \; q_j \rightarrow p_{j+1}\) in \(\Delta\) \((1 \leq i < m, \; 1 \leq j < n)\) implies \(S \xrightarrow{a} T \text{ in } \Delta_d\) does not hold in general. Why?
Example

Consider $A = (\{a, b\}, \{p, q\}, p, \{q\}, \Delta)$ where

- $\{a, b\}$ : alphabet
- $\{p, q\}$ : state symbols ($p$: initial state, $q$: final state)
- $\Delta$ : $p \xrightarrow{a} p$, $p \xrightarrow{q} q$, $b \xrightarrow{q}$

Step (1)
Add the transition rules

- $p \xrightarrow{a} q$ as $p \xrightarrow{a} p$ & $p \xrightarrow{q}$
- $p \xrightarrow{b} q$ as $p \xrightarrow{q}$ & $q \xrightarrow{b}$

Since no more transition rules can be added, all epsilon rules can be eliminated.

Step (2)
Define $Q_d = \{\{p\}, \{q\}, \{p, q\}\}$ (* $\emptyset$ (empty set) is eliminated for optimization and $\Delta_d$ to be

- $\{p\} \xrightarrow{a} \{p, q\}$
- $\{p, q\} \xrightarrow{a} \{p, q\}$
- $\{p\} \xrightarrow{b} \{q\}$
- $\{q\} \xrightarrow{b} \{q\}$
- $\{p, q\} \xrightarrow{b} \{q\}$

Since $\varepsilon$ is accepted by $A$, $\{p\}$ is also the final state, so $Q_{d_{\text{fin}}} = \{\{p\}, \{q\}, \{p, q\}\}$

Let $A_d = (\{a, b\}, Q_d, \{p\}, Q_{d_{\text{fin}}}, \Delta_d)$, then one can show that $L(A_d) = L(A)$

Closure properties

Let

- $C(FA_\Sigma)$ : set of languages over $\Sigma$ accepted by finite automata
- $op_n$ : $n$-ary function $2^{\Sigma^*} \times \cdots \times 2^{\Sigma^*} \rightarrow 2^{\Sigma^*}$

where $2^{\Sigma^*}$ means the set of all subsets of $\Sigma^*$

e.g. the example for $op_n$:

- $op_1$ : $(\cdot)^c$ (complement)
- $op_2$ : $\cup$ (union) $\cap$ (intersection) $\cdot$ (concatenation)

For all languages $L_1, \ldots, L_n$ in $C(FA_\Sigma)$, if $op_n(L_1, \ldots, L_n) \in C(FA_\Sigma)$, we say “the class $C(FA_\Sigma)$ is closed under $op_n$”

E.g. $C(FA_\Sigma)$ is closed under union iff for all $L_1, L_2$ in $C(FA_\Sigma)$, $L_1 \cup L_2$ in $C(FA_\Sigma)$.
The class $C(\text{FA}_\Sigma)$ is closed under union, intersection, complement

**Proof for $\cup$**

Suppose $A_1 = (\Sigma, P, p_0, P_{\text{fin}}, \Delta_1)$ and $A_2 = (\Sigma, Q, q_0, Q_{\text{fin}}, \Delta_2)$ are finite automata whose sets $P, Q$ of state symbols are disjoint to each other. Let $r_0$ be a fresh state symbol, then define $B = (\Sigma, P \cup Q \cup \{r_0\}, r_0, P_{\text{fin}} \cup Q_{\text{fin}}, \Delta_1 \cup \Delta_2 \cup \{r_0 \to p_0, r_0 \to q_0\})$. By construction, trivially $B$ accepts a word $w$ if and only if $A_1$ or $A_2$ accepts $w$.

**Proof for $\cap$**

From the previous proposition about DFA, we suppose $A_1 = (\Sigma, P, p_0, P_{\text{fin}}, \Delta_1)$ and $A_2 = (\Sigma, Q, q_0, Q_{\text{fin}}, \Delta_2)$ are already DFA. Define $C = (\Sigma, P \times Q, (p_0, q_0), P_{\text{fin}} \times Q_{\text{fin}}, \Delta)$ where

\[
P \times Q : \{\langle p, q \rangle \mid p \in P, q \in Q\}
\]

\[
P_{\text{fin}} \times Q_{\text{fin}} : \{\langle p, q \rangle \mid p \in P_{\text{fin}}, q \in Q_{\text{fin}}\}
\]

\[
\Delta : \{\langle p_1, q_1 \rangle \xrightarrow{a} \langle p_2, q_2 \rangle \mid p_1 \xrightarrow{a} p_2 \in \Delta_1, q_1 \xrightarrow{a} q_2 \in \Delta_2\}\}
\]

$C$ is DFA and it simulates the transition moves of $A_1$ and $A_2$, simultaneously.

**Proof for $\cap$ (another version)**

Suppose $A_1 = (\Sigma, P, p_0, P_{\text{fin}}, \Delta_1)$ and $A_2 = (\Sigma, Q, q_0, Q_{\text{fin}}, \Delta_2)$ are finite automata whose sets $P, Q$ of state symbols are disjoint. Let $r, s$ be fresh state symbols, then define $C' = (\Sigma, P \times Q, (p_0, q_0), P_{\text{fin}} \times Q_{\text{fin}}, \Delta')$ where

\[
\Delta' : \{\langle p_1, q_1 \rangle \xrightarrow{a} \langle p_2, q_2 \rangle \mid p_1 \xrightarrow{a} p_2 \in \Delta_1, q_1 \xrightarrow{a} q_2 \in \Delta_2\}\} \cup
\]

\[
\{\langle p_1, q \rangle \xrightarrow{a} \langle p_2, q \rangle \mid p_1 \xrightarrow{a} p_2 \in \Delta_1, q \in Q\}\} \cup
\]

\[
\{\langle p, q_1 \rangle \xrightarrow{a} \langle p, q_2 \rangle \mid q_1 \xrightarrow{a} q_2 \in \Delta_2, p \in P\}\}
\]

$C'$ simulates the transition moves of $A_1$ and $A_2$, simultaneously, whenever $\Delta_1$ contains $p_1 \xrightarrow{a} p_2$ and $\Delta_2$ contains $q_1 \xrightarrow{a} q_2$. For the transition move by $p_1 \xrightarrow{a} p_2$ of $A_1$, the other move in $A_2$ is suspended by assuming that for each state $q$ in $Q$, there exists a transition rule $q \xrightarrow{a} q$ in $\Delta_2$. For the move by $q_1 \xrightarrow{a} q_2$ of $A_2$, we assume a similar condition for $A_1$.

**Proof for $(\cdot)^c$**

The basic idea to construct a finite automaton that accepts for a given finite automaton $A$ the complement of $L(A)$ is similar to the construction of DFA from finite automata. See 6–7 in Exercise.
Example

Consider \( A_1 = (\{ a \}, P, p_0, \{ p_0 \}, \Delta_1) \) and \( A_2 = (\{ a \}, Q, q_0, \{ q_0 \}, \Delta_2) \)

where
\[
\begin{align*}
\Delta_1 & : \quad p_0 \xrightarrow{a} p_1 \quad p_1 \xrightarrow{a} p_0 \\
\Delta_2 & : \quad q_0 \xrightarrow{a} q_1 \quad q_1 \xrightarrow{a} q_2 \quad q_2 \xrightarrow{a} q_0
\end{align*}
\]

Note that
\[
\begin{align*}
\mathcal{L}(A_1) & = (aa)^* \quad \text{sequence of a's of multiple of 2} \\
\mathcal{L}(A_2) & = (aaa)^* \quad \text{sequence of a's of multiple of 3}
\end{align*}
\]

Now we define by product construction that
\[
\begin{align*}
Q_\times & : \quad (p_0, q_0) \ (p_0, q_1) \ (p_0, q_2) \ (p_1, q_0) \ (p_1, q_1) \ (p_1, q_2) \\
Q_\times\text{fin} & : \quad (p_0, q_0) \\
\Delta_\times & : \quad (p_0, q_0) \xrightarrow{a} (p_1, q_1) \quad (p_0, q_1) \xrightarrow{a} (p_1, q_2) \quad (p_0, q_2) \xrightarrow{a} (p_1, q_0) \\
& \quad (p_1, q_0) \xrightarrow{a} (p_0, q_1) \quad (p_1, q_1) \xrightarrow{a} (p_0, q_2) \quad (p_1, q_2) \xrightarrow{a} (p_0, q_0)
\end{align*}
\]

Let \( A_\times = (\{ a \}, Q_\times, \{(p_0, q_0)\}, Q_\times\text{fin}, \Delta_\times) \), then one can show \( \mathcal{L}(A_\times) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2) \)

Decidability

Computable function (by Turing 1936):

program, which is purely mechanical process, that for some of the input for an instance of a given problem, terminates and gives the correct yes/no answer * computable function is total if it is defined for every input

Cf. effective method by Rosser 1939

effectively calculable function by Kleene 1952 (Church 1936 for informal use)
effective procedure by Minsky 1967

but, algorithm is named after Abu Abdullah Muhammad ibn Musa al-Khwarizmi 825

Decidable problem:

A decision problem (Entscheidungsproblem) is called decidable if there exists a total computable function which solves the problem

Undecidable problem:

If there does not exist a total computable function solving the problem, the problem is called undecidable.
Proposition

The following problems are decidable for the class of finite automata:

- \( w \in \mathcal{L}(A) ? \) \hspace{1cm} (membership problem)
- \( \mathcal{L}(A) = \emptyset ? \) \hspace{1cm} (emptiness problem)
- \( \mathcal{L}(A) = \Sigma^* ? \) \hspace{1cm} (universality problem)
- \( \mathcal{L}(A) \subseteq \mathcal{L}(B) ? \) \hspace{1cm} (inclusion problem)
- \( \mathcal{L}(A) = \mathcal{L}(B) ? \) \hspace{1cm} (equivalence problem)

Proof

Omitted (proofs in a more general framework will be found in seminar talk 3). Note that if the emptiness problem is decidable, the other problems are decidable, due to the property that for finite automata \( A, B \), one can construct finite automata, each of which accepts the union \( \mathcal{L}(A) \cup \mathcal{L}(B) \), the intersection \( \mathcal{L}(A) \cap \mathcal{L}(B) \), the complement \( \overline{\mathcal{L}(A)} \). For instance, the membership problem can be rephrased to the question if \( \{w\} \cap \mathcal{L}(A) \neq \emptyset \).

Finite state diagrams

directed graph with

- \( \Sigma \) : alphabet
- \( Q \) : finite set of vertices with labels
- \( q_0 \) : start vertex such that \( q_0 \) in \( Q \)
- \( Q_{\text{fin}} \) : final vertices such that \( Q_{\text{fin}} \subseteq Q \)
- \( \Delta \) : \( \Sigma \)-labeled or unlabeled arrows connecting vertices

e.g. this finite state diagram accepts \( a^\ell b^m c^n \) \( (\ell, m, n \geq 0) \)

Cf. Moore machine, Mealy machine
regular grammar $G = (\Sigma, T, N, q_0, \Delta)$

- $\Sigma$ : alphabet
- $T$ : set of terminal symbols such that $T \subseteq \Sigma$
- $N$ : set of non-terminal symbols such that $N = \Sigma - T$
- $q_0$ : start symbol such that $q_0 \in N$
- $\Delta$ : finite set of production rules with the following forms
  
  \[
  p \rightarrow a \quad p \rightarrow aq \quad p \rightarrow \varepsilon \quad (p, q \in N, \ a \in T)
  \]

- word generated by $G$ : word over $T$ reachable from $q_0$
- language generated by $G$ : set of words generated by $G$
  (called regular language)

### Exercise

1. Can we determine if a given finite set is an alphabet or not? Specifically, given words $w_1, \ldots, w_n$ over an alphabet $\Sigma$, is it decidable whether \{ $w_1, \ldots, w_n$ \} is an alphabet?

2. [McMillan’s Theorem] Let $w_1, \ldots, w_n$ be $n$ non-empty words over the alphabet $\Sigma = \{ a_1, \ldots, a_k \}$. If \{ $w_1, \ldots, w_n$ \} is an alphabet, then

\[
\sum_{i=1}^{n} k^{-|w_i|} \leq 1
\]

where $|w_i|$ is the length of word $w_i$. Prove the above statement.

3. Let $\Sigma = \{ a_1, \ldots, a_k \}$ be the alphabet. Show that for $n$ natural numbers $p_1, \ldots, p_n$ that possibly include the same numbers, there exist $n$ non-empty words $w_1, \ldots, w_n$ over $\Sigma$ of length $p_1, \ldots, p_n$, respectively, such that \{ $w_1, \ldots, w_n$ \} is the alphabet if and only if

\[
\sum_{i=1}^{n} k^{-|w_i|} \leq 1.
\]
Exercise (cont’d)

4. Show that for every finite set \( L \) of words over an alphabet \( \Sigma \), one can construct a finite automaton \( A \) over \( \Sigma \) that accepts \( L \).

5. For a finite automaton \( A_d \) obtained from \( A \) in page 10, show that \( A_d \) is DFA and \( L(A_d) = L(A) \).

6. Even if the condition “\( T \neq \emptyset \) for \( \Delta_d \)” in the step (2) is eliminated, \( B \) is DFA and \( L(B) = L(A) \). Verify this statement.

7. Let \( D = (\Sigma, 2^Q, \{q_0\}, 2^Q - Q_{dfin}, \Delta_d) \) associated to the above \( B \) in 6. Show that \( L(D) = \Sigma^* - L(A) \).

8. Show that the class \( C(FA_{\Sigma}) \) is closed under concatenation. The concatenation of language \( L_1 \) to language \( L_2 \) over the same alphabet, denoted \( L_1 \cdot L_2 \), is defined as \( L_1 \cdot L_2 = \{ uv \mid u \in L_1, w \in L_2 \} \).

9. Show that languages accepted by finite automata are regular languages, and regular languages are accepted by finite automata.

Appendix: Post’s correspondence problem (PCP)

Given an alphabet \( \Sigma \)

instance of PCP of size \( n \) : \( n \) pairs of words \( v_i, w_i \) (\( i \leq n \)) over \( \Sigma \)

solution to this instance of : sequence \( i_1, i_2 \ldots i_k \) of indices

length \( k \) such that \( v_{i_1}v_{i_2}\ldots v_{i_k} \equiv w_{i_1}w_{i_2}\ldots w_{i_k} \)

Question if there is a solution to a given instance is undecidable [1], even for size 7 [2], but decidable for 2 [3]. Decidability for size 3–6 is unknown so far.


Advanced topics: MSO and regular languages

vocabulary \((F,R)\) of second-order logic:
- \(F\): a finite set of function symbols (with arity)
- \(R\): a finite set of relation symbols (with arity)

variables \((V_1,V_2)\):
- \(V_1\): a set of first-order variables
- \(V_2\): a set of second-order variables (with non-zero arity)

terms \(T\) ::= \(x\) if \(x \in V_1\)
- \(f(T,\ldots,T)\) if \(f \in F\)

formulas \(\Psi\) ::= \(T = T\)
- \(r(T,\ldots,T)\) if \(r \in R\)
- \(\exists x \, \Psi\)  \(\forall x \, \Psi\) if \(x \in V_1\)
- \(X(T,\ldots,T)\) if \(X \in V_2\)
- \(\exists X \, \Psi\)  \(\forall X \, \Psi\)
- \(\Psi \lor \Psi\)  \(\Psi \land \Psi\)  \(\neg \Psi\)

Examples

1. Given a finite set \(A\), define an SO-sentence \(\psi_1\) such that \(|A|\) is even if \(\psi_1\) has a model \(\mathcal{A}\) whose carrier is \(A\). We take \(\psi_1\) to be
   \[
   \exists X \, \exists F \left[ \forall x \forall y \left( F(x,y) \Rightarrow X(x) \land \neg X(y) \right) \land
   \forall x \exists y F(x,y) \land
   \forall x \forall y \exists x F(x,y) \land
   \forall x_1 \forall x_2 \forall y_1 \forall y_2 \left( F(x_1,y_1) \land F(x_2,y_2) \Rightarrow (y_1 = y_2 \Leftrightarrow x_1 = x_2) \right) \right]
   \]
   Then, \(\psi_1\) has a model \(\mathcal{A}_1\) whose carrier is \(A_1\) iff \(|A_1|\) is even (\(\vdash F\) is a bijective mapping from \(X\) to \(A_1 - X\)).

2. Graph connectivity: Given a finite directed graph \(G\) whose predicate \(e(\_\_\_)\) indicates the edge relation, the SO-sentence \(\psi_2\) defined below satisfies that \(G\) is connected iff \((G,\emptyset,\{e\}) \models \psi_2:\n   \forall X \neg\left[ \exists x X(x) \land \exists x \neg X(x) \land \forall x \forall y \left( X(x) \land \neg X(y) \Rightarrow \neg e(x,y) \right) \right]

3. Graph 3-colorability: Similarly, given a finite directed graph \(G\), we define the SO-sentence \(\psi_3\) below such that \(G\) is 3-colorable iff \((G,\emptyset,\{e\}) \models \psi_3:\n   \exists X \exists Y \exists Z \left[ \forall x \left\{ \bigvee_{U \in \{X,Y,Z\}} U(x) \Leftrightarrow \bigwedge_{V \in \{X,Y,Z\} - \{U\}} \neg V(x) \right\} \right] \land
   \forall x \forall y \left\{ \bigwedge_{U \in \{X,Y,Z\}} \left( e(x,y) \land U(x) \Rightarrow \neg U(y) \right) \right\]
SO vs. FO

SO is strictly more expressive than FO (first-order logic).

Proof

FO $\subseteq$ SO is obvious; we show below that the inclusion is strict, by using the following two theorems:

[Compactness] A set $S$ of sentences (i.e. closed formulas) over a vocabulary $(F,R)$ has a model iff every finite subset of $S$ has a model.

[Downward Löwenheim-Skolem] If a set $S$ of sentences over a vocabulary $(F,R)$ has a infinite model, then $S$ has a countable model.

Given a vocabulary $(F,R)$ whose $F$ and $R$ are empty, consider the $(F,R)$-structure $A = (A,\varnothing,\varnothing)$, where $A$ is the carrier. Suppose, for leading to contradiction, that there exists an FO-sentence $\psi$ such that $A \models \psi$ iff $|A|$ mod 2 = 0. Note that this property is definable in SO. Let $\phi_k = \exists x_1 \ldots \exists x_k \bigwedge_{i \neq j} \neg (x_i = x_j)$ for all $k \geq 0$, and define two sets of sentences, $S_1 = \{\psi\} \cup \{\phi_k \mid k \geq 0\}$ and $S_2 = \{\neg \psi\} \cup \{\phi_k \mid k \geq 0\}$. By Compactness theorem, each of $S_1$ and $S_2$ has a model. Let $A_i$ be a model of $S_i$ ($i \in \{1,2\}$). Since each model must be infinite, by Downward Löwenheim-Skolem theorem, $A_i = (A_i,\varnothing,\varnothing)$ such that $A_i$ is a countable set ($i \in \{1,2\}$). Thus, $A_1$ and $A_2$ are isomorphic. However, $A_1 \models \psi$ and $A_2 \models \neg \psi$, leading to the contradiction. $\square$

Monadic second-order logic (MSO)

$\text{MSO} : \forall X \in V_2 : X \text{ is 1-ary}$

$\exists \text{MSO} : \forall \psi \in \Psi : \psi \text{ is MSO formula } \exists X_1 \ldots \exists X_n \phi \text{ such that }$ 

$\phi \text{ does not contain } \exists X \text{ or } \forall X$

$\forall \text{MSO} : \forall \psi \in \Psi : \psi \text{ is MSO formula } \forall X_1 \ldots \forall X_n \phi \text{ such that }$ 

$\phi \text{ does not contain } \exists X \text{ or } \forall X$

Note 1

$\text{FO} \subseteq \exists \text{MSO}, \forall \text{MSO} \subseteq \text{MSO} \subseteq \text{SO}$

It is not obvious, however, whether each inclusion is strict or not, though $\text{FO} \not\subseteq \text{SO}$.

Note 2

- “Graph connectivity” is definable in $\forall \text{MSO}$ (Example 2 in page 23).
- “Graph 3-colorability” is definable in $\exists \text{MSO}$ (Example 3 in page 23).
SO-definable languages

For each word \( w \) over \( \Sigma \), define
\[
A_w : \text{finite structure } (\{1, \ldots, |w|\}, \emptyset, \{<_w\} \cup \{p_c(\cdot)\}_{c \in \Sigma})
\]
where \(< : \text{linear order on } \mathbb{N}\)
\[
p_c : p_c(i) = \text{true if } w = ucv (u, v \in \Sigma^*) \land |uc| = i
\]
(e.g. if \( w = aabcb \), then \( p_a(1) = \text{true} \), but \( p_b(1) = \text{false} \)).

A language defined by an SO-sentence \( \psi \), denoted \( L(\psi) \):
\[
L(\psi) = \{ w \in \Sigma^* | A_w \models \psi \}
\]

Example

Consider \( \psi = \exists X \left[ \forall x (\forall y (x = y \lor y < x) \Rightarrow X(x)) \land 
\forall x (\forall y (y = x \lor x < y) \Rightarrow \neg X(x)) \land
\forall x \forall y \{x < y \land \exists z (x < z \land z < y) \Rightarrow X(x) \iff \neg X(y)\} \right] \)
then
\[
L(\psi) = (ab)^*, (ba)^*, \ldots \text{ over the alphabet } \{a, b\}
\]

Büchi-Elgot-Trakhtenbrot’s theorem

A language is definable in MSO iff the language is regular.

Proof

First we show the “if” part: Given a DFA \( A = (\Sigma, Q, q_0, Q_{\text{fin}}, \Delta) \), we define below an (3)MSO-formula \( \psi_A \) such that \( L(\psi_A) = L(A) \). Let \( Q = \{q_0, \ldots, q_n\} \). We introduce SO-variables \( X_{q_0}, \ldots, X_{q_n} \) to indicate a state of the tape-head such that \( X_q(x) = \text{true} \) iff the head is in position \( q \). So, at the beginning, if \( q_0 \xrightarrow{c} q \), then
\[
\psi_{\text{init}} = \forall x \bigwedge_{c \in \Sigma} [p_c(x) \land \forall y (x = y \lor x < y) \Rightarrow X_q(x)].
\]
For each transition \( p \xrightarrow{c} q \), we have
\[
\psi_{\text{tran}} = \forall x \forall y \bigwedge_{p \in Q} \bigwedge_{c \in \Sigma} [X_p(x) \land p_c(x) \land (x < y \land \exists z (x < z \land z < y)) \Rightarrow X_q(y)].
\]
When accepting an input by the transition \( p \xrightarrow{c} q \) (\( q \in Q_{\text{fin}} \)),
\[
\psi_{\text{fin}} = \forall x [\forall y (x = y \lor y < x) \Rightarrow \bigvee_{q \in Q_{\text{fin}}} X_q(x)].
\]
We take the conjunction of the above three formulas to be \( \psi_A (X_{q_0}, \ldots, X_{q_n}) \). Then, by construction, it is not difficult to see \( L(\exists X_{q_0} \ldots \exists X_{q_n} \psi_A (X_{q_0}, \ldots, X_{q_n})) = L(A) \).

Next, to prove the “only if” part, we use a result obtained as a consequence of MSO Ehrenfeucht-Faissé theorem (e.g. [4]). The result is explained in the next page.

Proof (cont’d)

Let \( \psi \) be an MSO-formula, we write \( \text{rank}(\psi) = k \) if the depth of quantifier nesting is at most \( k \) \((k \in \mathbb{N})\), and we write \( \text{MSO}[k] \) for the set \( \{ \psi \mid \text{rank}(\psi) \leq k \} \). Given two structures \( A \) and \( B \), we say \( A \) and \( B \) are elementary MSO-equivalent up to \( k \), denoted \( A \equiv_{k}^{\text{MSO}} B \), when \( A \models \psi \) iff \( B \models \psi \) for all \( \psi \in \text{MSO}[k] \). Then it is known that the following property hold:

**Lemma 1** For every vocabulary \((F_0, R)\), where \( F_0 \) is a finite set of constant symbols, with finite fixed numbers \( m, n \) of MSO and FO free variables, \( \text{MSO}[k] \) can be partitioned to \( S_1, \ldots, S_\ell \) and contains formulas \( \psi_1(x_m, \bar{x}_n), \ldots, \psi_\ell(x_m, \bar{x}_n) \), such that

1. for every structure \( A \) with the carrier \( A \), and elements \( \bar{a}_m \in A^m \) and \( \bar{U}_n \in (2^A)^n \), there exists \( i \) such that \( A \models \psi_i(\bar{a}_m, \bar{U}_n) \) iff \( A \models \theta(\bar{a}_m, \bar{U}_n) \) for all \( \theta \in S_i \),
2. for every \( \theta \in \text{MSO}[k] \), there exists \( J \) such that \( \theta \) is equivalent to \( \bigvee_{j \in J} \psi_j \).

We suppose that a language \( L \) over \( \Sigma \) is defined by an MSO sentence \( \delta \) with \( \text{rank}(\delta) = k \). According to the above lemma, over a vocabulary \((\emptyset, \{<\} \cup \{p_i\}_{i \in \Sigma})\) with no SO or FO free variable, \( \text{MSO}[k] \) can be partitioned to \( S_1, \ldots, S_\ell \), and \( \text{MSO}[k] \) contains sentences \( \psi_1, \ldots, \psi_\ell \) that satisfy the above conditions 1 and 2. This implies that there exists a subset \( F \) of \( \{1, \ldots, \ell\} \) such that \( \delta \equiv \bigvee_{i \in F} \psi_i \). Moreover, there exists some \( S_e \) that contains a sentence logically equivalent to \( \neg \exists x (x = x) \). Now we define the finite automaton \( A_\delta = (\Sigma, \{1, \ldots, \ell\}, e, F, \Delta) \).

Proof (cont’d)

The transition rules in \( \Delta \) of \( A_\delta \) are defined as follows:

\[ i \xrightarrow{c} j \quad \text{if} \quad B_w \models \theta \quad \text{for all} \quad \theta \in S_i \quad \text{and} \quad B_{wc} \models S_j \quad \text{for all} \quad \theta \in S_j. \]

We show below that for every word \( w \), after reading \( w \), the automaton \( A_\delta \) ends in some state \( i \) \((1 \leq i \leq \ell)\) such that \( B_w \models \theta \) for all \( \theta \in S_i \). We use induction on the length of \( w \). The base case is obvious, because \( A_\delta \) ends in the initial state \( e \). For induction step, we suppose \( w = wc \) for some \( c \in \Sigma \). By induction hypothesis, after reading \( w \), \( A_\delta \) is in state \( i \) and \( B_w \models \theta \) for all \( \theta \in S_i \). If the next character on the tape is \( c \), there is some \( j \) such that \( B_{wc} \models \theta \) for all \( \theta \in S_j \). That means, \( A_\delta \) ends in such \( j \) after reading \( wc \). Note that \( \exists x (x = x) \) (\( \equiv \) true) is in \( \text{MSO}[k] \). Finally, we consider the language accepted by \( A_\delta \). From the above observation, the language \( L(A_\delta) \) is \( \{ w \in \Sigma^* \mid \exists i \in F : B_w \models \psi_i \} \), which is equivalent to \( \{ w \in \Sigma^* \mid \exists i \in F : B_w \models \psi_i \} \). Hence, because \( \delta \equiv \bigvee_{i \in F} \psi_i \), we have \( L(\delta) = L(A_\delta) \).

**Corollary**

\( \text{MSO} = \exists \text{MSO} \) over words.

Proof

According to the proof of the previous theorem, every language definable in MSO is regular, and every regular language is expressible in \( \exists \text{MSO} \).
There exists a language definable in SO, but not in MSO.

**Proof**

We show that the language \( L = \{ w \in \{ a, b \}^* \mid |w|_a = |w|_b \} \) is definable in SO. Here \(|w|_a\) means the number of occurrences of \( a \) in \( w \). Define \( \psi \) to be

\[
\exists X \forall F \left[ \forall x (X(x) \leftrightarrow p_a(x) \land \neg X(x) \leftrightarrow p_b(x)) \land \forall x \forall y (F(x, y) \Rightarrow X(x) \land \neg X(y)) \land \forall x \exists y F(x, y) \land \forall y \exists x F(x, y) \land \forall x_1 \forall x_2 \forall y_1 \forall y_2 (F(x_1, y_1) \land F(x_2, y_2) \Rightarrow (y_1 = y_2 \iff x_1 = x_2)) \right].
\]

The above sentence \( \psi \) specifies that for each word \( w \) over the alphabet \( \{a, b\} \), the number of occurrences of \( a \) and \( b \) in \( w \) are the same iff \( \psi \). Thus, \( L(\psi) = L \).

However, \( L \) is not a regular language, which will be explained later in Exercises 5,6 in the next seminar talk. Indeed, \( L \) is a context-free language (See page 5, seminar talk 2). Therefore, by Büchi-Elgot-Trakhtenbrot’s theorem (page 27), \( L \) is not definable in MSO.

**Note**

Over the alphabet \( \Sigma = \{a\} \), a language is context-free iff it is definable in MSO. (Cf. Exercise 7, seminar talk 2)

There exists a language definable in MSO, but not in FO.

**Proof**

In order to show the statement, one should introduce the following result :

[Gurevich, 1984] Given two linear ordered structures \( A = (A, \emptyset, \{<_A\} \cup R_A) \) and \( B = (B, \emptyset, \{<_B\} \cup R_B) \), where \( <_X \) is a linear order over \( X \ (X \in \{A, B\}) \), then \( |A|, |B| \geq 2^k \) implies \( A \equiv ^{\text{FO}} B \).

Using the above property, we show that \( (aa)^* \) is not definable in FO. Suppose, for leading to contradiction, that there exists an FO-sentence \( \psi \) such that \( L(\psi) = (aa)^* \) and \( \text{rank}(\psi) = k \). Let \( A_w \) be a structure associated to a word \( w \) in \( a^* \), and let \( B_{wa} \) be a structure associated to \( wa \). Obviously, \( A_w \models \psi \iff B_{wa} \models \psi \). However, according to Gurevich’s claim, if \( |w| \geq 2^k \), then \( A_w \equiv ^{\text{FO}} k B_{wa} \), leading to the contradiction.

**Corollary**

There is no FO-sentence \( \psi_{\text{even}} \) such that for every linear ordered structure \( A = (A, \emptyset, \{<\} \cup R), |A| \) is even iff \( A \models \psi_{\text{even}} \).

(∵ If \( \psi_{\text{even}} \) exists, the language \( (aa)^* \) is definable by \( \psi_{\text{even}} \).)
1. Show that the Hamiltonicity of finite undirected graphs can be expressed in SO, i.e. construct an SO-sentence $\psi_{\text{Ham}}$ over graph structures such that a finite undirected graph $G$ is Hamiltonian iff $G \models \psi_{\text{Ham}}$. (Note that a finite undirected graph is Hamiltonian iff there exists a path in the graph which visits every vertex exactly once.)

2. Show that $a^*b^*$ is definable in FO, i.e. construct an FO-sentence $\psi$ such that $\mathcal{L}(\psi) = a^*b^*$. Likewise, show that $\Sigma^*$ and $\emptyset$ are definable in FO.

3. Construct an MSO-sentence that defines $a^*(bb)^*a^*$.

4. Show that there is no FO-sentence $\psi_{\text{conn}}$ such that for every finite directed graph $G = (G, \emptyset, \{e\})$, $G$ is connected iff $G \models \psi_{\text{conn}}$. (Hint: Use the previous corollary guaranteeing that there is no FO-sentence which expresses the linear ordered set is even.)

5. Show that $\text{FO} \subsetneq \exists \text{MSO}$ and $\text{FO} \subsetneq \forall \text{MSO}$. 

32
II. Grammar
Grammar

grammar : $G = (\Sigma, T, N, q_0, \Delta)$

$\Sigma$ : alphabet
$T$ : set of terminal symbols such that $T \subseteq \Sigma$
$N$ : set of non-terminal symbols such that $N = \Sigma - T$
$q_0$ : start symbol such that $q_0 \in N$
$\Delta$ : finite set of production rules with the following forms

$\alpha \rightarrow \beta \quad (\alpha, \beta \in \Sigma^*)$

Cf. regular grammar if production rules are in the following forms:

$p \rightarrow a \quad p \rightarrow a q \quad p \rightarrow \varepsilon \quad (p, q \in N, \ a \in T)$

Generated languages

Given grammar $G = (\Sigma, T, N, q_0, \Delta)$, define for words $u, w$ over $\Sigma$,

$u \rightarrow^*_G w : \exists \alpha \rightarrow \beta$ in $\Delta$ such that

- $u$ is decomposed to words $u_1, \alpha, u_2$
- $w$ is decomposed to words $u_1, \beta, u_2$

$u_0 \rightarrow^*_G u_n : \text{if } u_0 \equiv u_n, \text{ or}$

- if $u_0 \rightarrow^*_G u_1 \& u_1 \rightarrow^*_G u_n$
  
  (the reflexive and transitive closure of $\rightarrow_G$)

$L(G) : \text{set of words over } T \text{ such that for all } w \text{ in } L(G), q_0 \rightarrow^*_G w$

(language generated by $G$)

Note

Question if $w \in L(G)$ is undecidable in general. (Cf. Rice’s theorem in seminar 5)
Decidable sub-classes

grammar whose production rules are in the following forms is called

context-sensitive : $\alpha \rightarrow \beta$  ($\alpha, \beta \in \Sigma^*$ such that $|\alpha| \leq |\beta|$, $q_0 \rightarrow \varepsilon$ start symbol $q_0$ never appears on right-hand side of any rule)

context-free : $p \rightarrow \beta$  ($p \in N$, $\beta \in \Sigma^*$)

language is called

context-sensitive if it is generated by context-sensitive grammar
context-free if it is generated by context-free grammar

Note

Membership problem, i.e. the question if $w \in L(G)$, is decidable for the classes of context-sensitive grammar and context-free grammar. Why? (Exercise)

Example

Define the production rules

$\Delta_1$ : $q_0 \rightarrow a q_0 b$  $q_0 \rightarrow \varepsilon$

$\Delta_2$ : $q_0 \rightarrow a q_1$  $q_0 \rightarrow b q_2$  $q_0 \rightarrow \varepsilon$

$q_1 \rightarrow a q_1 q_1$  $q_1 \rightarrow b q_0$

$q_2 \rightarrow a q_0$  $q_2 \rightarrow b q_2 q_2$

for the grammar

$G_1 = (\Sigma_1, \{a, b\}, \{q_0\}, q_0, \Delta_1)$

$G_2 = (\Sigma_2, \{a, b\}, \{q_0, q_1, q_2\}, q_0, \Delta_2)$

then

$L(G_1) = \{a^n b^n | n \geq 0\}$  $L(G_2) = \{w \in \{a, b\}^* | |w|_a = |w|_b\}$

where $|w|_a$ (resp. $|w|_b$) is the number of occurrences of $a$ ($b$) in $w$.
Eliminating $\varepsilon$-derivations

Given grammar $G = (\Sigma, T, N, q_0, \Delta)$, for non-terminal symbol $q$ in $N$, $q$ is called **nullable** if $q$ admits $q \rightarrow^* G \varepsilon$.

**Claim 1**

One can compute the set $N_\varepsilon$ of nullable non-terminal symbols if $G$ is context-free grammar.

**Claim 2**

If $G$ is context-free grammar, define $G' = (\Sigma, T, N, q_0, \Delta')$ where

$$\Delta' = \{ q \rightarrow \alpha_0 \beta_1 \alpha_1 \cdots \beta_n \alpha_n \mid \exists p \rightarrow \alpha_0 p_1 \alpha_1 \cdots p_n \alpha_n \in \Delta : \exists \alpha_i \in (T \cup N - N_\varepsilon)^*, \exists p_1, \ldots, p_n \in N_\varepsilon, \beta_i \in \{ p_i, \varepsilon \} \text{ (} 1 \leq i \leq n \text{) } \}$$

Then

$$L(G') = L(G) - \{ \varepsilon \} \quad \text{... show this claim (Exercise)}$$

**Proof of Claim 1**

We show that for each $q \in N$, $q$ is nullable if and only if $q \in N_\varepsilon$, where $N_\varepsilon$ is obtained by the following procedure. This procedure halts on any context-free grammar.

$$N_\varepsilon := \emptyset$$

while $S = \emptyset$ do

$$S := \{ q \in N - N_\varepsilon \mid \exists q \rightarrow \alpha \in \Delta \text{ such that } \alpha \in N_\varepsilon^* \} ;$$

$$N_\varepsilon := N_\varepsilon \cup S$$

od

return $N_\varepsilon$

The “if” part is shown by induction on the number $n$ of loops. For induction step, let $S_n$ be the set $S$ obtained at the $n$-th loop ($n \geq 1$). For $q \in S_n$, there is $q \rightarrow \alpha$ with $\alpha \in N_\varepsilon^*$. By induction hypothesis, non-terminals of $\alpha$ are all nullable, so $\alpha \rightarrow^* G \varepsilon$, and hence $q \rightarrow^*_G \varepsilon$.

The “only if” part is shown by the length of $q \rightarrow^*_G \varepsilon$. If the length is 1, $\Delta$ contains $q \rightarrow \varepsilon$, and thus, $q$ is added to $N_\varepsilon$ at 1st loop. For induction step, suppose $q \rightarrow_G \alpha$ and $\alpha \rightarrow^*_G \varepsilon$ whose length is $n$ ($n \geq 1$) and $\alpha \in N_\varepsilon^*$. By assumption, $\Delta$ contains $q \rightarrow \alpha$. If $q \notin N_\varepsilon$, $q \in S$ at some loop, because it satisfies that $q \in N - N_\varepsilon$, $q \rightarrow \alpha \in \Delta$ and $\alpha \in N_\varepsilon^*$. Hence, $q \in N_\varepsilon$. □
Proposition

For context-free grammar \( G \) with the set \( T \) of terminal symbols, one can construct context-free grammar \( G' = (\Sigma, T, N, q_0, \Delta') \) such that \( L(G') = L(G) \) and \( \Delta' \) contains transition rules with the following forms only:

\[
\begin{align*}
q & \to \alpha \\
q_0 & \to \varepsilon
\end{align*}
\]

\((q \in N, \alpha \in (\Sigma - \{q_0\})^+)\),

\(q_0\) never appears in the right-hand side of any rule.

Proof

This is an easy consequence of the previous Claims 1,2 except the treatment of the start symbol and the production rule of the form \( q_0 \to \varepsilon \). First, replace the start symbol, say \( p_0 \), of \( G \) by \( q_0 \) and add \( q_0 \to p_0 \) to the set of production rules of \( G \). After constructing \( G' \), test if \( G \) generates \( \varepsilon \), which is the decidable problem for context-free grammar. If yes, then add \( q_0 \to \varepsilon \) to \( \Delta' \) of \( G' \).

Corollary

Context-free languages are generated by context-sensitive grammar. (This is not obvious only from grammar definitions on page 4, but from the above Proposition.)

Pumping lemma [Bar-Hillel 1961]

Given context-free language \( L \),

\[
\exists k \geq 0 : \text{if } z \in L \text{ and } |z| \geq k, \text{ then } z \text{ is formed by } u, v, w, x, y
\]

as

\[
\begin{array}{cccccc}
  u & v^n & w^n & x^n & y \\
\end{array}
\]

such that \(|vx| \geq 1\) and \( uv^nwx^n y \in L \) \( (n \geq 0) \)

Proof

Suppose \( G = (\Sigma, T, N, q_0, \Delta) \) is context-free grammar whose production rules are already in the forms of \( q_0 \to \varepsilon \) and \( q \to \alpha \) for \( \alpha \in (\Sigma - \{q_0\})^+ \). For \( p, q \in \Sigma \), we say \( p \) is a descendant of \( q \) if \( q \to^* p \) upw. If the length of the derivation \( q \to^* p \) upw is more than 0, \( p \) is a proper descendant of \( q \). Observe that the (proper) descendant relation is closed under contexts, i.e. \( p \) is a descendant of \( q \) such that \( q \to^* p \) upw if and only if for all \( s, t \) in \( \Sigma^* \), \( sqt \to^* s upw t \). For every word \( z \) generated by \( G \), one can construct a directed graph of the proper descendant relation associated to \( z \), whose node is marked by an element in \( \Sigma \) and the edge is the proper descendant relation obtained from the derivation \( q_0 \to^* z \).

(proof cont’d)
Proof (cont’d)

If $G$ generates a word $z$ whose directed graph of the proper descendant relation admits the path starting from $q_0$ to $\alpha$ in $T$ such that the length of this path is more than $|N|$, then the path contains a sub-path from $p$ to $p$. (This situation corresponds to the figure on the right.) If the yellow part is removed, $q_0 \rightarrow^*_G u w y$; if this part is repeated $n$-times, it turns out $q_0 \rightarrow^*_G u v^n w x^n y$. Since $p$ is the proper descendant of $p$ in this case, by the shape of rules, $|v x| \geq 1$.

Proposition 1

Given context-free grammar $G$, question if $L(G) = \emptyset$ is decidable.

Proof

Suppose $G = (\Sigma, T, N, q_0, \Delta)$, each of whose rules is already in the form of $q_0 \rightarrow \varepsilon$ or $q \rightarrow \alpha$ for $q \in N$, $\alpha \in (\Sigma - \{q_0\})^*$. Let $\ell = \max\{|\alpha| \mid p \rightarrow \alpha \in \Delta\}$ and $k = \ell|N|+1$, then define $L_{\leq k} = \{w \in T^* \mid |w| \leq k\}$. From the proof of the previous lemma, $L_{\leq k} = \emptyset$ if and only if $L(G) = \emptyset$. Because $L_{\leq k}$ is finite, question if $L_{\leq k} = \emptyset$ is decidable.

Example

Using Pumping Lemma, we show that $L = \{a^\ell b^\ell a^\ell \mid \ell \geq 1\}$ is not context-free. For leading the contradiction, we suppose below that $L$ is context-free.

Take $z = a^k b^k a^k$ in $L$. If $k$ is sufficiently large, $z$ can be decomposed to $u, v, w, x, y$ such that $|v x| \geq 1$ and $u v^n w x^n y \in L$ for all $n \geq 0$.

If $|v| = 0$, then $x = a^i$ or $x = b^i$ with $i \leq k$. However, $u v^2 w x^2 y$ can not be an element in $L$. Thus, $|v| \neq 0$.

If $|v| \geq 1$, then $v = a^i$ or $v = b^i$ with $i \leq k$. If $|x| = 0$, then it immediately leads to the contradiction, so $|x| \neq 0$.

If $|x| \geq 1$ also, then $x = a^j$ or $x = b^j$ with $j \leq k$. However, in any combination of the possibility of $v$ and $x$, $u v^2 w x^2 y$ can not be an element in $L$, leading to the contradiction.
Proposition 2

Given two context-free grammars $G_1, G_2$, the intersection-emptiness problem, i.e. the question if $L(G_1) \cap L(G_2) = \emptyset$, is undecidable.

Proof

Use the reduction from PCP. For an instance of PCP $I = \{\langle u_1, w_1 \rangle, \ldots, \langle u_n, w_n \rangle\}$ over $\Sigma$, define the grammar $G_I = (\Sigma \cup \{q_0\}, \Sigma, \{q_0\}, q_0, \Delta_I)$, provided that $q_0 \notin \Sigma$:

$$
\Delta_I : q_0 \rightarrow \overline{u_1}q_0w_1 \ldots q_0 \rightarrow \overline{u_n}q_0w_n \quad q_0 \rightarrow \epsilon
$$

where $\overline{u_i}$ denotes the reverse of word $u_i$, e.g. $\overline{abcd} = dcba$. Moreover, if $\Sigma = \{a_1, \ldots, a_k\}$, define the grammar $G_{eq} = (\Sigma \cup \{q_0, q_1\}, \Sigma, \{q_0, q_1\}, q_0, \Delta_{eq})$, where

$$
\Delta_{eq} : q_0 \rightarrow a_1q_1a_1 \ldots q_0 \rightarrow a_kq_1a_k
$$

$$
q_1 \rightarrow a_1q_1a_1 \ldots q_1 \rightarrow a_kq_1a_k \quad q_1 \rightarrow \epsilon
$$

Note that $L(G_{eq})$ does not contain the empty word $\epsilon$. Then, by construction, $L(G_I) \cap L(G_{eq}) \neq \emptyset$ if and only if $I$ has a solution. However, question if an instance of PCP has a solution is undecidable.

Closure properties (1)

Let $C(CF_T)$ be the set of context-free languages over $T$ of terminal symbols

Proposition

The class $C(CF_T)$ is closed under union. However, the class $C(CF_T)$ is not closed under intersection or complement if $|T| \geq 2$

Proof

For union, let $G_1 = (\Sigma_1, T, N_1, p_0, \Delta_1)$ $G_2 = (\Sigma_2, T, N_2, q_0, \Delta_2)$ be context-free grammar. We suppose $N_1 \cap N_2 = \emptyset$. Let $r_0$ be fresh symbol, then define $G = (\Sigma_1 \cup \Sigma_2 \cup \{r_0\}, T, N_1 \cup N_2 \cup \{r_0\}, r_0, \Delta_1 \cup \Delta_2 \cup \{r_0 \rightarrow p_0, r_0 \rightarrow q_0\})$. By construction, for word $w \in T^*$, $G$ generates $w$ if and only if $G_1$ or $G_2$ generates $w$.

For not being closed under intersection, it suffices to show that $L_1 = \{a^\ell b^m a^m | \ell, m \geq 1\}$ and $L_2 = \{a^m b^\ell a^\ell | \ell, m \geq 1\}$ are generated by context-free grammar.

For not being closed under complement, use de Morgan’s law, because $L_1 \cap L_2 = \overline{(L_1)^c \cup (L_2)^c)}$. 

□
Closure properties (2)

Let \( C(\text{CS}_T) \) be the set of context-sensitive languages over \( T \) of terminal symbols

**Proposition**

The class \( C(\text{CS}_T) \) is closed under union, intersection, complement

**Proof**

For union, one can apply the same proof of the previous proposition.

For intersection, let \( G_1 = (\Sigma_1, T, N_1, p_0, \Delta_1) \) and \( G_2 = (\Sigma_2, T, N_2, q_0, \Delta_2) \) be context-sensitive grammars. Assume that production rules in \( G_1, G_2 \) are Kuroda normal form except the rules \( p_0 \rightarrow \varepsilon, q_0 \rightarrow \varepsilon \) if each of them exists in \( \Delta_1 \) or \( \Delta_2 \), respectively. This assumption does not lose the generality of the discussion. (See Exercise.)

Define the grammar \( G = (\Sigma, T, N, (p_0, q_0), \Delta) \). The set \( N \) of non-terminal symbols is defined below, and production rules in \( \Delta \) are defined in the next slide.

\[
N : N_1 \cup \{ (p, q) \mid p \in N_1, q \in N_2 \}
\]

(proof cont’d)

**Proof (cont’d)**

\[
\Delta : \quad (p_1, q_0) \rightarrow (q_1, q_0) r_1 \quad \text{if} \quad p_1 \rightarrow q_1 r_1 \text{ in } \Delta_1 \\
(p_1, q_0, q_1) \rightarrow (r_1, q_0) s_1 \quad \text{if} \quad p_1 q_1 \rightarrow r_1 s_1 \text{ in } \Delta_1 \\
(p_1, q_0) \rightarrow (q_1, q_0) \quad \text{if} \quad p_1 \rightarrow q_1 \text{ in } \Delta_1 \\
(p_1, q_0) q_1 \rightarrow p_1 (q_1, q_0) \quad \text{if} \quad p_1, q_1 \text{ in } N_1 \\
p_1 (q_1, q_0) \rightarrow (p_1, q_0) q_1 \quad \text{if} \quad p_1, q_1 \text{ in } N_1 \\
(p_1, p_2) q_1 \rightarrow (p_1, q_2) (q_1, r_2) \quad \text{if} \quad p_1, q_1 \text{ in } N_1, \quad p_2 \rightarrow q_2 r_2 \text{ in } \Delta_2 \\
(p_1, p_2) (q_1, q_2) \rightarrow (p_1, r_2) (q_1, s_2) \quad \text{if} \quad p_1, q_1 \text{ in } N_1, \quad p_2 q_2 \rightarrow r_2 s_2 \text{ in } \Delta_2 \\
(p_1, p_2) \rightarrow (p_1, q_2) \quad \text{if} \quad p_1 \text{ in } N_1, \quad p_2 \rightarrow q_2 \text{ in } \Delta_2 \\
(p_1, p_2) \rightarrow a \quad \text{if} \quad p_1 \rightarrow a \text{ in } \Delta_1, \quad p_2 \rightarrow a \text{ in } \Delta_2 \\
(p_0, q_0) \rightarrow \varepsilon \quad \text{if} \quad p_0 \rightarrow \varepsilon \text{ in } \Delta_1, \quad q_0 \rightarrow \varepsilon \text{ in } \Delta_2
\]

One can show that for each \( p_1, p_2, \ldots, p_n \) in \( N_1 \) and \( q_1, q_2, \ldots, q_n \) in \( N_2 \),

\[
p_0 \rightarrow^{*}_G p_1 p_2 \cdots p_n \text{ iff } (p_0, q_0) \rightarrow^{*}_G (p_1, q_0) p_2 \cdots p_n
\]

\[
q_0 \rightarrow^{*}_G q_1 q_2 \cdots q_n \text{ iff } (p_1, q_0) p_2 \cdots p_n \rightarrow^{*}_G (p_1, q_1) (p_2, q_2) \cdots (p_n, q_n)
\]

**Corollary**

For the class of \( C(\text{CS}_T) \), question if \( L(G) = \emptyset \) is undecidable

(Cf. The intersection-emptiness problem for \( C(\text{CF}_T) \) on page 12)
1. Show that for the class of context-sensitive grammar, membership problem is decidable.

2. Show Claim 2.

3. **Chomsky normal form** if production rules are in the following forms:
   \[ p \rightarrow qr \quad p \rightarrow a \quad q_0 \rightarrow \varepsilon \quad (p \in N, q, r \in N - \{q_0\}, a \in T) \]
   Show that a language is generated by a grammar in Chomsky normal form if and only if it is generated by a context-free grammar.

4. A grammar \( G = (\Sigma, T, N, q_0, \Delta) \) is called **Kuroda normal form** if production rules are in the following forms:
   \[ pq \rightarrow rs \quad p \rightarrow qr \quad p \rightarrow q \quad p \rightarrow a \quad (p, q, r, s \in N, a \in T) \]
   Show that languages generated by grammar in Kuroda normal form are context-sensitive, and conversely, context-sensitive languages which does not include the empty word are generated by grammar in Kuroda normal form.

**Exercise (cont’d)**

5. [Pumping lemma for regular grammar] Given regular language \( L \), there exists \( k \geq 0 \) such that if \( z \in L \) and \( |z| \geq k \), then \( z \) is formed by \( u, v, w \) with \( |v| \geq 1 \) and \( uv^nw \in L \) \((n \geq 0)\). Verify this statement.

6. Show that the language \( L = \{ a^\ell b^\ell \mid \ell \geq 0 \} \) is not generated by regular grammar.

7. Show that context-free languages over a singleton set of a symbol are generated by regular grammar.

8. For each of languages \( L_1 = \{ a^\ell b^\ell a^m \mid \ell, m \geq 1 \} \) and \( L_2 = \{ a^m b^\ell a^\ell \mid \ell, m \geq 1 \} \), construct context-free grammar, respectively.

9. [Liu & Weiner 1973] For each \( n \geq 0 \), let \( C(CF_i^n_T) \) be the class of languages over \( T \) defined as follows: \( C(CF_0^T) \) is the set of context-free languages whose set of terminal symbols is \( T \); \( C(CF_{i+1}^T) \) is \( \{ L_1 \cap L_2 \mid L_1 \in C(CF_i^T), L_2 \in C(CF_i^T) \} \). Show that for all \( i \geq 0 \), \( C(CF_i^T) \subset C(CF_{i+1}^T) \).

10. Show that \( C(CF_i^n_T) \subset C(CS_T) \) for all \( n \geq 0 \).
**Appendix: Linear-bounded automata (LBA)**

LBA is a non-deterministic Turing machine with rewritable length-bounded single-tape as input: \((\Sigma \cup \{\sharp_L, \sharp_R\}, Q, q_0, Q_{\text{fin}}, \Delta)\), where \(
\sharp_L, \sharp_R\) are special tape symbols meaning left- right-ends. Transition rules in \(\Delta\) are in the form of \((p, a) \rightarrow (q, b, x)\) \((p, q \in Q, a, b \in \Sigma, x \in \{L, R\})\).

\[
\begin{align*}
\bullet & q \\
\downarrow & (q, d, L) \leftarrow (p, b) \\
\sharp_L \ldots a \ d \ c \ldots \sharp_R \\
\downarrow & p \\
\downarrow & (p, b) \rightarrow (r, e, R) \\
\downarrow & (r, e, R) \\
\sharp_L \ldots a \ e \ c \ldots \sharp_R \\
\end{align*}
\]

The class of languages accepted by linear-bounded automata, denoted by \(\text{NSPACE}(n)\), and \(C(\text{CS}_T)\) coincide [1]. From space complexity observation, it can be shown that \(\text{NSPACE}(n) = \text{co-NSPACE}(n)\) [2]. This implies that for all \(L \in C(\text{CS}_T)\), \((L)^c \in C(\text{CS}_T)\), stating that \(C(\text{CS}_T)\) is closed under complement.

---


III. Trees
Signature and trees

Given a finite set $F = \{ f_0, \ldots, f_n \}$ of symbols

$F$ is a **signature** if $F$ is equipped with mapping $\text{ar}_F : F \to \mathbb{N}$

$f$ has **arity** $i$ if $\text{ar}(f) = i$

$F(i) = \{ f \in F | \text{ar}_F(f) = i \}$

The set of (finite) **trees** over $F$, denoted by $T_F$:

$f(t_1, \ldots, t_n) \in T_F$ if $f \in F(n)$ & $t_1, \ldots, t_n \in T_F$ ($n \geq 0$)

$f()$ is simply denoted by $f$, called **constant**

**Note**

By definition, there is no empty tree, though the empty word exists

---

Positions

Let $p$ be a word over $\mathbb{N} - \{0\}$

$p$ is a **position** of $t$ ($t \in T_F$)

if $p = \varepsilon$ or

$t = f(t_1, \ldots, t_n)$ & $p = iq$ ($1 \leq i \leq n$) & $q$ is a position of $t_i$

$\text{pos}(t) :$ set of positions of $t$ ($\varepsilon$ is root position)

$\text{size}(t) :$ the number of elements in pos($t$)

$\text{height}(t) : \max\{ |w| | w \in \text{pos}(t) \}$

**Example**

Let $t$ be the tree $f(g(a), f(b, c))$

$\text{pos}(t) = \{ \varepsilon, 1, 11, 2, 21, 22 \}$

$\text{size}(t) = 6$

$\text{height}(t) = 2$
Contexts and substitutions

Let □ be a fresh constant symbol, called hole

\[ C_F : \text{set of trees over } F \cup \{\square\}, \text{ called contexts} \]

\[ C[\ ] : \text{context with one } \square \text{ in } C \]

\[ C[t] : \text{tree whose } \square \text{ in } C[\ ] \text{ is replaced by } t \]

Let \( V \) be set of symbols with \( F \cap V = \emptyset \) & \( \text{ar}_{F \cup V}(x) = 0 \ (x \in V) \)

\[ T_{F,V} : \text{set } T_{F \cup V} \text{ of trees with variables} \]

\[ \theta : \text{mapping } V \to T_{F,V}, \text{ called assignment} \]

\[ t \theta : \theta(x) \text{ if } x \in V ; f(t_1 \theta, \ldots, t_n \theta) \text{ if } t = f(t_1, \ldots, t_n) \]

( homomorphic extension† of \( \theta \), called substitution )

Let \( R \) be binary relation over \( T_{F,V} \)

\[ R \text{ is closed under contexts} : s \; R \; t \Rightarrow \; C[s] \; R \; C[t] \]

\[ R \text{ is closed under substitutions} : s \; R \; t \Rightarrow \; s \theta \; R \; t \theta \]

Tree automata (TA)

tree automaton \((F, Q, Q_{\text{fin}}, \Delta)\)

\[ F : \text{signature} \]

\[ Q : \text{finite set of state symbols} \]

\[ Q_{\text{fin}} : \text{finite set } Q_{\text{fin}} (\subseteq Q) \text{ of final states} \]

\[ \Delta : \text{finite set of transition rules with the following forms} \]

\[ f(p_1, \ldots, p_n) \to q \quad \text{[regular rule]} \]

\[ p \to q \quad \text{[epsilon rule]} \]

for \( f \in F_{(n)}, p_1, \ldots, p_n, p, q \in Q \)

regular tree automaton (RTA) if \( \Delta \) does not contain an epsilon rule

Note

RFA (= epsilon-rule free FA) & FA \(\propto\) RTA & TA
Tree automata vs. finite automata

Tree automata

input

transition rules

Automata

Accepted trees

Given a tree automaton $A = (F, Q, Q_{\text{fin}}, \Delta)$

$\rightarrow_A$ (move relation): $s \rightarrow_A t$ if $\exists l \rightarrow r \in \Delta$, $C[\ ] \in C_{F \cup Q}$:

$s = C[l] \& t = C[r]$

$t$ is accepted by $A$: $\exists t \in T_F, q \in Q_{\text{fin}}: t \rightarrow_A \cdots \rightarrow_A q$

($A$ accepts $t$)

tree language $L$: $L \subseteq T_F$

tree language $L(A)$: set of trees accepted by $A$

$A$ accepts $L$: $L(A) = L$

regular tree language: tree language accepted by a regular TA
Consider \( \mathcal{A} = (\{0, 1, \lor, \land, \neg\}, Q, Q_{\text{fin}}, \Delta) \) where

\[
\begin{align*}
Q & : q_0, q_1 \\
Q_{\text{fin}} & : q_1 \\
\Delta & : 0 \rightarrow q_0 \quad 1 \rightarrow q_1 \quad \neg(q_0) \rightarrow q_1 \quad \neg(q_1) \rightarrow q_0 \quad \lor(q_0, q_0) \rightarrow q_0 \quad \lor(q_0, q_1) \rightarrow q_1 \quad \lor(q_1, q_0) \rightarrow q_1 \quad \lor(q_1, q_1) \rightarrow q_1 \quad \land(q_0, q_0) \rightarrow q_0 \quad \land(q_0, q_1) \rightarrow q_0 \quad \land(q_1, q_0) \rightarrow q_0 \quad \land(q_1, q_1) \rightarrow q_1
\end{align*}
\]

We take \( \land(\lor(0, 1), \land(1, 0)) \):

\[
\begin{align*}
\land(\lor(0, 1), \land(1, 0)) & \rightarrow_{\mathcal{A}} \land(\lor(q_0, 1), \land(1, 0)) \\
& \rightarrow_{\mathcal{A}} \land(\lor(q_0, q_1), \land(1, 0)) \\
& \rightarrow_{\mathcal{A}} \land(\lor(q_0, q_1), \land(q_1, 0)) \\
& \rightarrow_{\mathcal{A}} \land(q_1, \land(q_1, q_0)) \\
& \rightarrow_{\mathcal{A}} \land(q_1, q_0) \\
& \rightarrow_{\mathcal{A}} q_0 \\
\land(\lor(0, 1), \land(1, 1)) & \rightarrow_{\mathcal{A}}^* q_1 \quad \text{accepted by } \mathcal{A}
\end{align*}
\]

### Deterministic tree automata (DTA)

TA \((F, Q, Q_{\text{fin}}, \Delta)\) is deterministic if

1. \(\Delta\) contains regular transition rules only (so it is a regular TA)
2. there are no rules \(f(p_1, \ldots, p_n) \rightarrow q_1\) and \(f(p_1, \ldots, p_n) \rightarrow q_2\) with \(q_1 \neq q_2\)

#### Proposition \((TA = DTA)\)

Given a TA \(\mathcal{A}\), one can construct a DTA \(\mathcal{B}\) such that \(L(\mathcal{A}) = L(\mathcal{B})\)

**Proof**

Similar to the case of finite automata (see page 8–9 in the first talk), the procedure for constructing DTA consists of the two steps: Let \(\mathcal{A} = (F, Q, Q_{\text{fin}}, \Delta)\),

1. eliminate epsilon rules from \(\Delta\)
2. eliminate non-determinisity by subset construction for the rules \(f(p_1, \ldots, p_n) \rightarrow q_1\) and \(f(p_1, \ldots, p_n) \rightarrow q_2\) with \(q_1 \neq q_2\)

Complete the proof in Exercise.\(\square\)
Consider \( A = (\{ f, a \}, Q, Q_{\text{fin}}, \Delta) \) where

\[
Q : q_a q \\
Q_{\text{fin}} : q \\
\Delta : q_a \rightarrow q \quad a \rightarrow q_a \quad f(q_a, q_a) \rightarrow q_a
\]

Step (1) Eliminate epsilon rules by adding transition rules

\[
a \rightarrow q \quad \text{as} \quad a \rightarrow q_a \quad q_a \rightarrow q \\
f(q_a, q_a) \rightarrow q \quad \text{as} \quad f(q_a, q_a) \rightarrow q_a \quad q_a \rightarrow q
\]

Step (2) Eliminate non-determinisity by subset construction (p.10 in seminar 1)

\[
Q_q : \{ q_a \} \{ q \} \{ q_a, q \} \quad \exists \emptyset \text{ is eliminated for optimization} \\
Q_{q_{\text{fin}}} : \{ q \} \{ q_a, q \} \\
\Delta_d : a \rightarrow \{ q_a, q \} \quad \text{as} \quad a \rightarrow q_a \land a \rightarrow q \\
\quad f(\{ q_a \}, \{ q_a \}) \rightarrow \{ q_a, q \} \quad f(\{ q_a, q \}, \{ q_a \}) \rightarrow \{ q_a, q \} \\
\quad f(\{ q_a \}, \{ q, q \}) \rightarrow \{ q, q \} \quad f(\{ q_a, q \}, \{ q, q \}) \rightarrow \{ q_a, q \} \\
\quad \text{as} \quad f(q_a, q_a) \rightarrow q_a \land f(q_a, q_a) \rightarrow q
\]

Let \( A_d = (\{ f, a \}, Q_d, Q_{d_{\text{fin}}}, \Delta_d) \), then one can show that \( L(A) = L(A_d) \)

---

**Closure properties**

Let \( C(TA_F) : \) the class of regular tree languages over \( F \)

**Proposition**

The class \( C(TA_F) \) is closed under union, intersection, complement

**Proof for \( \cup \)**

Suppose \( A_1 = (F, P, P_{\text{fin}}, \Delta_1) \) and \( A_2 = (F, Q, Q_{\text{fin}}, \Delta_2) \) are tree automata whose sets \( P, Q \) of state symbols are disjoint. Define \( B = (F, P \cup Q, P_{\text{fin}} \cup Q_{\text{fin}}, \Delta_1 \cup \Delta_2) \), then \( L(B) = L(A_1) \cup L(A_2) \). One should note that \( t \rightarrow_\delta \cdots \rightarrow_\delta q \ (q \in P_{\text{fin}} \cup Q_{\text{fin}}) \) if and only if \( t \rightarrow_{A_1} \cdots \rightarrow_{A_1} q \ (q \in P_{\text{fin}}) \) or \( t \rightarrow_{A_2} \cdots \rightarrow_{A_2} q \ (q \in Q_{\text{fin}}) \), whose derivation consists of only \( \rightarrow_{A_1} \) or \( \rightarrow_{A_2} \).

**Proof for \( \cap \)**

Suppose \( A_1 = (F, P, P_{\text{fin}}, \Delta_1) \) and \( A_2 = (F, Q, Q_{\text{fin}}, \Delta_2) \) are tree automata. First, construct the tree automata \( A'_1 = (F, P, P_{\text{fin}}, \Delta'_1) \) and \( A'_2 = (F, Q, Q_{\text{fin}}, \Delta'_2) \) such that \( L(A_1) = L(A'_1) \) and \( L(A_2) = L(A'_2) \) and there is no epsilon rules in \( \Delta'_1 \) and \( \Delta'_2 \). This transformation can be done by the procedure (1) on page 9. (Proof cont’d)
Proof for $\cap$ (cont’d)

Define $\mathcal{C} = (F, P \times Q, P_{\text{fin}} \times Q_{\text{fin}}, \Delta)$ as follows.

$P \times Q : \{ (p, q) \mid p \in P, q \in Q \}$

$P_{\text{fin}} \times Q_{\text{fin}} : \{ (p, q) \mid p \in P_{\text{fin}}, q \in Q_{\text{fin}} \}$

$\Delta : \{ f((p_1, q_1), \ldots, (p_n, q_n)) \rightarrow (p, q) \mid f(p_1, \ldots, p_n) \rightarrow p \in \Delta'_1 \}$

$\Delta_{\text{fin}} : \{ f(q_1, \ldots, q_n) \rightarrow q \}$

$\mathcal{C}$ simulates $\mathcal{A}_1$ and $\mathcal{A}_2$ move by move, simultaneously. This means that $s \rightarrow^c t$ using the rule $f((p_1, q_1), \ldots, (p_n, q_n)) \rightarrow (p, q)$ if and only if there exists the corresponding moves $s_1 \rightarrow A_1$, $t_1$ by $f(p_1, \ldots, p_n) \rightarrow p$ and $s_2 \rightarrow A_2$, $t_2$ by $f(q_1, \ldots, q_n) \rightarrow q$, where $s_i$ $(1 \leq i \leq 2)$ is obtained from $s$ by taking $i$-th projection of the product states. This observation implies that $s \rightarrow^c \ldots \rightarrow^c (p, q)$ $(s, q) \in P_{\text{fin}} \times Q_{\text{fin}}$ if and only if $s \rightarrow A_1$, $\ldots \rightarrow A_1$, $p (p \in P_{\text{fin}})$ and $s \rightarrow A_2$, $\ldots \rightarrow A_2$, $q (q \in Q_{\text{fin}})$. \hfill \square

Proof for $(\cdot)^C$

Using the subset construction, define $D = (F, 2^Q, \{ p \in 2^Q \mid p \cap Q_{\text{fin}} \neq \emptyset \}, \Delta')$, where for each $p_1, \ldots, p_n \in 2^Q, f(p_1, \ldots, p_n) \rightarrow p \in \Delta'$ if $p = \{ q \in Q \mid \exists q_i \in p_i \ (1 \leq i \leq n) : f(q_1, \ldots, q_n) \rightarrow q \in \Delta \}$. Note that $\mathcal{D}$ is deterministic and completely defined (which means for each $p_1, \ldots, p_n \in 2^Q$, there exists $f(p_1, \ldots, p_n) \rightarrow p$ in $\Delta'$). Moreover, $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{D})$ (Exercise). Let $D' = (F, 2^Q, \{ p \in 2^Q \mid p \cap Q_{\text{fin}} = \emptyset \}, \Delta')$, then $\mathcal{L}(D') = (\mathcal{L}(D))^C$. \hfill \square

Example

Consider $\mathcal{A}_1 = (\{ f, a, b \}, \{ p \}, \{ p \}, \Delta_1)$ and $\mathcal{A}_2 = (\{ f, a, b \}, \{ q, q_a, q_b \}, \{ q \}, \Delta_2)$ where

$\Delta_1 : a \rightarrow p \quad b \rightarrow p \quad f(p, p) \rightarrow p$

$\Delta_2 : a \rightarrow q_a \quad b \rightarrow q_b \quad q_a \rightarrow q_b \quad f(q_a, q_b) \rightarrow q \quad f(q_b, q_a) \rightarrow q$

Note that

$\mathcal{L}(\mathcal{A}_1) = T_{\{ f, a, b \}}$

$\mathcal{L}(\mathcal{A}_2) = \{ f(a, b), f(b, a) \}$

Define by product construction (p.14 in seminar 1) that

$Q_{\times} : (p, q_a) (p, q_b) (p, q)$

$Q_{\times\text{fin}} : (p, q)$

$\Delta_{\times} : a \rightarrow (p, q_a) \quad b \rightarrow (p, q_b) \quad f((p, q_a), (p, q_b)) \rightarrow (p, q) \quad f((p, q_b), (p, q_a)) \rightarrow (p, q)$

superimpose transition rules of $\Delta_1$ onto ones in $\Delta_2$

Let $\mathcal{A}_{\times} = (\{ f, a, b \}, Q_{\times}, Q_{\times\text{fin}}, \Delta_{\times})$, then one can show that $\mathcal{L}(\mathcal{A}_{\times}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.
Example of ($L(A)$)\(^C\)

Consider $A = (\{f, a\}, \{q, q_a\}, \{q\}, \Delta)$ where

$$\Delta: a \rightarrow q_a \quad f(q_a, q_a) \rightarrow q$$

Note that

$L(A) = \{f(a, a)\}$

Define $A' = (\{f, a, b\}, Q', Q_{\text{fin}}, \Delta')$ where

$$Q': \emptyset \{q\} \{q_a\} \{q, q_a\}$$

$Q_{\text{fin}}: \emptyset \{q_a\}$

$\Delta': a \rightarrow \{q_a\}$

1. $f(\emptyset, \emptyset) \rightarrow \emptyset$  
   $f(\emptyset, \{q\}) \rightarrow \emptyset$

2. $f(\emptyset, \{q_a\}) \rightarrow \emptyset$  
   $f(\emptyset, \{q, q_a\}) \rightarrow \emptyset$

3. $f(\{q\}, \emptyset) \rightarrow \emptyset$  
   $f(\{q\}, \{q\}) \rightarrow \emptyset$

4. $f(\{q\}, \{q_a\}) \rightarrow \emptyset$  
   $f(\{q\}, \{q, q_a\}) \rightarrow \emptyset$

5. $f(\{q_a\}, \emptyset) \rightarrow \emptyset$  
   $f(\{q_a\}, \{q\}) \rightarrow \emptyset$

6. $f(\{q, q_a\}, \emptyset) \rightarrow \emptyset$  
   $f(\{q, q_a\}, \{q\}) \rightarrow \emptyset$

7. $f(\{q, q_a\}, \{q_a\}) \rightarrow \{q\}$  
   $f(\{q, q_a\}, \{q, q_a\}) \rightarrow \{q\}$

8. $f(\{q, q_a\}, \{q\}) \rightarrow \emptyset$  
   $f(\{q, q_a\}, \{q, q_a\}) \rightarrow \{q\}$

9. $f(\{q, q_a\}, \{q, q_a\}) \rightarrow \{q\}$  
   $f(\{q, q_a\}, \{q, q_a\}) \rightarrow \{q\}$

Pumping lemma for TA

Given regular tree language $L$,

$$\exists k \geq 0: \text{ if } t \in L \text{ & } \text{height}(t) \geq k$$

then $t = C[D[u]]$

such that $\text{size}(D) > 1$ \& $C[D^n[u]] \in L \ (n \geq 0)$

\textbf{Proof}

Recall the proof of \textbf{Pumping lemma for CFG} (pages 9-10 in seminar talk 2). Suppose $L = L(A)$ and $A = (F, Q, Q_{\text{fin}}, \Delta)$ is regular. If $t \in L(A)$ and $\text{height}(t) \geq \min(|Q|, |\Delta|)$, then $t \rightarrow_A^* q$ ($q \in Q$) implies that $t = C[D[u]]$ such that $u \rightarrow_A^* p$ and $D[p] \rightarrow_A^* p$ and $C[p] \rightarrow_A^* q$ for some $p \in Q$. Since $D^n[p] \rightarrow_A^* p$ ($n \geq 0$), we have $C[D^n[u]] \rightarrow_A^* q$. \hfill \Box
Example

Consider $A_1 = (\{ f, a \}, \{ q, q_a \}, \{ q \}, \Delta_1)$ and $A_2 = (\{ f, a \}, \{ q, q_a \}, \{ q \}, \Delta_2)$ where 

$\Delta_1 : a \rightarrow q_a \quad f(q_a, q_a) \rightarrow q$

$\Delta_2 : a \rightarrow q_a \quad f(q_a, q_a) \rightarrow q \quad f(q_a, q_a) \rightarrow q$

Let $k_1 = \min(\{q, q_a\}, \Delta_1) = 2$

There is no tree $t \in L(A_1)$ such that $\text{height}(t) \geq k_1$

No tree needed to be decomposed as the height of trees in $L(A_1)$ is $< k_1$

Let $k_2 = \min(\{q, q_a\}, \Delta_2) = 2$

For every tree $t \in L(A_2)$ such that $\text{height}(t) \geq k_2$, e.g., $f(f(a, a), a)$:

\[
\begin{array}{c}
\begin{array}{c}
 \text{f} \\
 q_a \\
 \end{array} \\
 \begin{array}{c}
 a \\
 q_a \\
 \end{array}
\end{array}
\]

we have $t = C[D[u]]$ such that $\text{size}(D) > 1$ & $C[D^n[u]]$ ($n \geq 0$)

Decidability

The following problems are decidable for the class of tree automata:

$t \in L(A)$? (membership problem)

$L(A) = \emptyset$? (emptiness problem)

$L(A) = T_F$? (universality problem)

$L(A) \subseteq L(B)$? (inclusion problem)

$L(A) = L(B)$? (equivalence problem)

Proof

Observe that the universality, inclusion, equivalence problems are special cases of the emptiness problem under the closure property of TA stating that for any TA $A, B$, one can construct tree automata, each of which accepts the union, intersection and complement. Moreover, according to the previous proof, $A = (F, Q, Q_{\text{fin}}, \Delta)$ accepts some tree (so $L(A) \neq \emptyset$) if and only if $A$ accepts a tree whose height is at most $\min(|Q|, |\Delta|)$. The set $T$ of trees with a bounded height is finite, because every symbol in $F$ has a fixed arity. So, if the membership problem is decidable, one can determine whether there exists a tree in $T$ that is accepted by $A$. The decidability of the membership problem is explained in the next proof.
Proof for the membership problem

Use König’s lemma (1936) stating that:

every finitely branching tree is finite if and only if it has only finite paths.

A tree with a finite number of branches at each fork and with a finite number of leaves at the end of each branch is called a finitely branching tree. Suppose $\mathcal{A} = (F, Q, Q_{\text{fin}}, \Delta)$ does not contain an epsilon rule, because it is possible to obtain an equivalent RTA from a given TA. Observe that for each move $s \rightarrow_A t$, the number of symbols in $F$ occurring in $s$ decreases. This implies that every sequence $t_1 t_2 \ldots$ of trees of $T_{F \cup Q}$ is finite if $t_i \rightarrow_A t_{i+1} (i \geq 1)$. Now we consider computation trees of $\mathcal{A}$ which are (possibly infinite) trees, and each of whose node is labeled by a tree in $T_{F \cup Q}$ such that nodes $s, t$ are connected by an edge from $s$ to $t$ if and only if $s \rightarrow_A t$. Then, by definition, every computation tree represents all possible trails of the computation by $\mathcal{A}$ (starting from some tree in $T_F$). Because $\mathcal{A}$ contains finitely many transition rules, all these computation trees are finitely branching. Moreover, because of the above observation, every path from the root toward a leaf (called run) is finite. Hence, every computation tree of $\mathcal{A}$ is finite according to König’s lemma. The membership problem for TA can then be rephrased to the question if a computation tree whose root node is labeled by a given tree in $T_F$ has a leaf of some state in $Q_{\text{fin}}$. Since the comprehensive search of a finite tree can be done by finitely many recursive calls, the membership problem for TA is decidable. 

Myhill-Nerode’s theorem (for words)

Given a language $L$ over $\Sigma$

Let $x \equiv_L y$ : binary relation over $\Sigma^*$ such that for all $z \in \Sigma^*$,

$xz \in L$ if and only if $yz \in L$ (called $L$-equivalent)

Proposition

The statements 1 & 2 are equivalent : for a language $L$ over $\Sigma$,

1. $L$ is regular
2. $\{ \{ x \in \Sigma^* \mid x \equiv_L w \} \mid \exists w \in \Sigma^* \} $ is finite

Proof

To show “1 ⇒ 2”, suppose that DTA $\mathcal{A} = (F, Q, q_0, Q_{\text{fin}}, \Delta)$ accepts $L$. Define $u \sim_A w$ if and only if $\mathcal{A}$ halts with the same state on $u$ and $w$. $A$ is not necessarily completely defined. Then, $u \sim_A w$ implies $u \equiv_L w$, because for any $z \in \Sigma^*$, if $u \sim_A w$, $\mathcal{A}$ halts with the same state on $uz$ and $wz$. Since the size of $\{ \{ x \in \Sigma^* \mid x \sim_A w \} \mid \exists w \in \Sigma^* \}$ is $|Q|$, the size of $\{ \{ x \in \Sigma^* \mid x \equiv_L w \} \mid \exists w \in \Sigma^* \} $ is at most $|Q|$. (Proof cont’d)
To show "2 $\Rightarrow$ 1", define $Q = \{ x \in \Sigma^* \mid x \approx_L w \} \mid \exists w \in \Sigma^* \}$, $q_0 = \{ x \in \Sigma^* \mid x \approx_L \epsilon \}$, $Q_{\text{fin}} = \{ x \in \Sigma^* \mid x \approx_L w \} \mid \exists w \in L \}$, $\Delta = \{ p \xrightarrow{a} q \mid p,q \in Q, a \in \Sigma : \exists x \in p \& xa \in q \}$. By assumption, $Q$ is finite, and $q_0$ must be an element in $Q$. Besides, $Q_{\text{fin}}$ is finite, because $Q$ is finite. Let $B = (\Sigma, Q, q_0, Q_{\text{fin}}, \Delta)$, then $B$ is completely defined DFA, because each $q \in Q$ is non-empty, and $p \neq q$ $(p,q \in Q)$ implies $p \cap q = \emptyset$. By case analysis of input, one can show that $L(B) = L$ : Obviously, $\epsilon \in L$ if and only if $q_0 \in Q_{\text{fin}}$. If $B$ reads $w$ and halts with the state $p$, then $wa (a \in \Sigma)$ is accepted by $B$ if and only if there exists a rule $p \xrightarrow{a} q$ in $\Delta$ with some $q$ in $Q_{\text{fin}}$. Because $x \approx_L y$ and $y \in L$ imply $x \in L$, $wa \approx_L u$ for some $u \in L$ (so $wa \in L$) if and only if $q \in Q_{\text{fin}}$. □

Corollary

For every regular language $L$ over $\Sigma$, the following statements are equivalent :

1. the number of elements in $\{ x \in \Sigma^* \mid x \approx_L w \}$ is $k$
2. the number of states of DFA that accepts $L$ is at least $k$

Proof

"1 $\Rightarrow$ 2" is an easy consequence of the previous proof. For the reverse, it suffices to show that $u \sim_A w$ if and only if $u \approx_L w$, provided that $A$ is minimal DFA. Here DFA $A$ is minimal if there is no DFA that accepts $L(A)$ and whose number of states is less than $A$’s. To complete the proof (showing that $u \sim_A w$ implies $u \not\approx_L w$) is

Exercise.

Myhill-Nerode’s theorem for trees

Given a tree language $L$ over $F$

Let $s \approx_L t$ : binary relation over $T_F$ such that for all $C \in C_F$,

$$C[s] \in L \text{ if and only if } C[t] \in L$$

Proposition

The statements 1 & 2 are equivalent : for a tree language $L$ over $F$, 1. $L$ is regular
2. $\{ s \in T_F \mid s \approx_L t \} \mid \exists t \in T_F \}$ is finite

Proof

To show "1 $\Rightarrow$ 2", similar to the previous proof, one can show that the size of the set $\{ s \in T_F \mid s \approx_L t \} \mid \exists t \in T_F \}$ is at most the number of state symbols of DTA that accepts $L$. To show "2 $\Rightarrow$ 1", define the TA $B = (F, Q, Q_{\text{fin}}, \Delta)$, where $Q = \{ s \in T_F \mid s \approx_L t \} \mid \exists t \in T_F \}$, $Q_{\text{fin}} = \{ s \in T_F \mid s \approx_L t \} \mid \exists t \in L \}$, $\Delta = \{ f(p_1,\ldots,p_n) \xrightarrow{q} \mid p_1,\ldots,p_n,q \in Q, f \in F : \exists i \in p_i (1 \leq i \leq n) \& f(t_1,\ldots,t_n) \in q \}$. One can show that for all $t \in T_F$, $t$ is accepted by $B$ if and only if $t \in L$. □
Exercise

1. Complete the proof on page 9.
2. Show the claim in the proof on page 11 that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{D})$.
3. Verify Corollary on page 20. (Complete the proof of “$2 \Rightarrow 1$”.)
4. Show that given a TA $\mathcal{A}$, one can construct minimal DTA that accepts $\mathcal{L}(\mathcal{A})$.
5. Define $\text{leaf}(t) = \text{leaf}(t_1) \cdots \text{leaf}(t_n)$ if $t = f(t_1, \ldots, t_n)$ and $n \geq 1$ ;
   $\text{leaf}(t) = t$ if $t \in F(0)$. Let $\mathcal{C}(\text{RLT}_F)$ be the class of tree languages
   over $F$ such that $L \in \mathcal{C}(\text{RLT}_F)$ if there exists a regular grammar $G$ with
   $L = \{ t \in T_F \mid \text{leaf}(t) \in \mathcal{L}(G) \}$. Show that (1) $\mathcal{C}(\text{RLT}_F) \subset
   \mathcal{C}(\text{TA}_F)$, and (2) $\mathcal{C}(\text{RLT}_F)$ is closed under Boolean operations.
6. Let $L$ be the tree language over $\{f, a, b\}$ such that $f(a, b) \in L$ ;
   $f(t_1, t_2) \in L$ if $t_1, t_2 \in L$. Show that $L$ is a regular tree language.
7. Let $F = \{f, a, b\}$ with $\text{ar}_F(f) = 2$ and $\text{ar}_F(a) = \text{ar}_F(b) = 0$, and $L =
   \{ t \in T_F \mid$ the numbers of occurrences of $a$ and $b$ in $t$ are the same$\}$. Show that $L$ is not a regular tree language.

Appendix : “On the Myhill-Nerode Theorem for Trees”

In late 1950’s, Nerode stated in [1] the well-known property originated
from Myhill’s early work [2]. After a while, Brainerd generalized this
result for automata on finite trees [3]. His generalized result, however,
seemed less accessible for those who are not familiar with universal
algebra or category theory. In 1977, Kozen showed an accessible pre-
sentation of the tree version of Myhill-Nerode’s theorem in his thesis
(without proof). Fülpü & Vágvölgyi also but independently showed
a similar result (with proof) in 1989. Kozen then provided a simple
proof and a simple notion of “tree automata” in [4] which is quite
acceptable for ones familiar with the original Myhill-Nerode’s theorem.

IV. Modulo Axioms
Set modulo relation

Let $\succsim$: binary relation on a set $S$

$\succsim$ is **preorder** if it is reflexive† and transitive‡ (†, ‡: next page)

$\succsim$ is **partial order** if it is reflexive and transitive and

[Anti-symmetric] $a \succsim b \& b \succsim a \Rightarrow a \equiv b$

$\succsim$ is defined as $\succsim - (\succsim)^{-1}$, called **strict part** of $\succsim$

$\succsim$ is **well-founded** if it does not admit an infinite sequence $s_1, s_2, s_3, \cdots$ such that $s_i \succsim s_{i+1}$ ($i \geq 1$)

$\text{min}_{\succsim}(S)$ is set of minimal elements in $S$ with respect to a well-founded preorder $\succsim$

$[a]_{\succsim}$ is set $\{x \in S | x \succsim a\}$

$S/\succsim$ is set $\{[a]_{\succsim} | \exists a \in \text{min}_{\succsim}(S)\}$

Equivalence classes

Let $\sim$: binary relation on a set $S$

$\sim$ is **equivalence relation** if for all $a, b, c$ in $S$,

[Reflexive] $a \sim a$

[Symmetric] $a \sim b \Rightarrow b \sim a$

[Transitive] $a \sim b \& b \sim c \Rightarrow a \sim c$

$(S, \sim)$ is called **setoid**

$[a]_{\sim}$ is set $\{x \in S | x \sim a\}$, **equivalence class** of $a$

$S/\sim$ is set of all equivalence classes $\{[a]_{\sim} | \exists a \in S\}$, **quotient set** of $S$ by $\sim$

Note

Every equivalence relation $\sim$ is a well-founded preorder, and $\text{min}_{\sim}(S) = S$ ($\because$ the strict part of $\sim$ is empty)
Let \( F \) : signature

\( F \)-algebra \( A = (C, \{ f_A \mid f \in F \}) \)

\( C \) : non-empty set, called carrier set of \( A \)

\( f_A \) : mapping from \( C^n \) (\( n \)-tuples of \( C \)) to \( C \) if \( \text{ar}_F(f) = n \)
(called operation of \( A \))

Let \( A = (C, \{ f_A \mid f \in F \}) \) and \( B = (D, \{ f_B \mid f \in F \}) \)

\( F \)-homomorphism \( h \) from \( A \) to \( B \)

\( h \) : mapping from \( C \) to \( D \) such that for all \( f \in F \),

if \( \text{ar}_F(f) = n \), then for all \( c_1, \ldots, c_n \in C \),

\( h(f_A(c_1, \ldots, c_n)) = f_B(h(c_1), \ldots, h(c_n)) \)

\( F \)-algebras

Congruence relations

Let \( \sim \) : binary relation on \( C \)

\( \sim \) is congruence relation of \( A = (C, \{ f_A \mid f \in F \}) \) if

- \( \sim \) is equivalence relation,
- for all \( f \in F \), \( f_A(c_1, \ldots, c_n) \sim f_A(d_1, \ldots, d_n) \)
  if \( \text{ar}_F(f) = n \) \& \( c_1 \sim d_1 \) \& \( \cdots \) \& \( c_n \sim d_n \)
  \( (c_i, d_i \in C \, , \, 1 \leq i \leq n) \)

\( A \) with \( \sim \) is quotient \( F \)-algebra, denoted by \( A/\sim \)

\( C/\sim \) : carrier set of \( A/\sim \)

\( f_A/\sim \) : set of operations of \( A/\sim \) defined as

\( (f_A/\sim)([c_1]_\sim, \ldots, [c_n]_\sim) = [f_A(c_1, \ldots, c_n)]_\sim \)
Equational theory

Let $F$ : signature

equation $s \approx t$ is a pair of trees $s, t$ with variables in $T_{F,V}$

equational theory $E = (F, E)$

$E$ : finite set of equations

Note

Equations $s \approx t$ and $t \approx s$ are distinguished if $s \not\equiv t$ (equations are orientation sensitive), and thus, $E_1 = \{ a \approx b \}$ and $E_2 = \{ a \approx b, b \approx a \}$ are not the same.

$\rightarrow_E$ : binary relation over $T_{F,V}$ such that $s \rightarrow_E t$ if

$\exists l \approx r \in E : s = C[l\theta] \& t = C[r\theta]$ for some $C[], \theta$

$\leftrightarrow_E$ : the smallest symmetric relation containing $\rightarrow_E$

$\equiv_E$ : the smallest equivalence relation containing $\rightarrow_E$

Example : AC-theory and multisets

Let $F = \{ f, a, b \}$ with $ar_F(f) = 2, \ ar_F(a) = ar_F(b) = 0$

$A = (T_F, \{ f_A(x,y) = f(x,y), \ a_A = a, \ b_A = b \})$

$B = (\text{mul} (\{a,b\}), \{ f_B(x,y) = x \cup \text{mul} y, \ a_B = \{a\}_{\text{mul}}, \ b_B = \{b\}_{\text{mul}} \})$

Define $F$-homomorphism $h$:

Then for all $s, t$ in $T_F$:

$h(s) = h(t)$ if and only if $[s]_{=E} = [t]_{=E}$

where $E = (F, E_{AC})$ and $E_{AC} = \{ f(x,y) \approx f(y,x), \ f(f(x,y),z) \approx f(x,f(y,z)) \}$

$\text{mul}(S)$ : set of multisets over $S$, $\cup_{\text{mul}}$ : multiset union, $\{\cdot\}_{\text{mul}}$ : multiset notation
Example: A-theory and words

Let $F = \{ f, a, b \}$ with $ar_F(f) = 2$, $ar_F(a) = ar_F(b) = 0$

$A = (T_F, \{ f_A(x,y) = f(x,y), a_A = a, b_A = b \})$

$B = (\{ a, b \}^*, \{ f_B(x,y) = x \cdot y, a_B = a, b_B = b \})$

Define $F$-homomorphism $h$:

\[ h(f_A(t_1, t_2)) = h(t_1) \cdot h(t_2) \]
\[ h(a_A) = a, \quad h(b_A) = b \]

then for all $s, t$ in $T_F$:

\[ h(s) = h(t) \text{ if and only if } [s]_{E_A} = [t]_{E_A} \]

where $E = (F, E_A)$ and $E_A = \{ f(f(x,y), z) \approx f(x, f(y,z)) \}$

$x \cdot y$: concatenation of words $x$ and $y$

$E$-equivalence problem

instance: trees $s, t$ in $T_F$
equational theory $E = (F, E)$
solution: “yes” if $[s]_E = [t]_E$; “no” otherwise

This problem is called $E$-equivalence problem

e.g. consider

$E_A : f(f(x,y), z) \approx f(x, f(y,z))$

$E_I : f(x,x) \approx x$

$E_C : f(x,y) \approx f(x,y)$

$E_U : f(x,e) \approx x$

then

the problem for equational theory $E$ with any combination of the above equations (16 combinations including the empty theory) is decidable.
Word problem

instance : trees $s, t$ in $T_F$
equational theory $E = (F, E \cup E_A)$, where
\[
E = \{ s_1 \approx t_1, \ldots, s_n \approx t_n \} \quad (s_i, t_i \in T_F, 1 \leq i \leq n)
\]
solution : “yes” if $[s]_E = [t]_E$; “no” otherwise

Note

According to the previous example, if $F = \{ f \} \cup F(0)$, it holds that
\[
[s]_E = [t]_E \text{ if and only if } \quad h(s) \text{ and } h(t) \text{ are equivalent under the axioms } h(s_1) \approx h(t_1), \ldots, h(s_n) \approx h(t_n)
\]
Words $u, w$ are equivalent under axioms $u_i \approx w_i$ ($u_i, w_i \in \Sigma^+, 1 \leq i \leq n$) if $u \equiv w$ or there exists a sequence $x_1 \ x_2 \ \cdots \ x_k$ ($k \geq 2$) such that (1) $x_1 = u, x_k = w$, (2) \[\exists u\ell_i \approx w\ell_i \quad (1 \leq i < k) : \quad x_i = y_i u\ell_i z_i \& x_{i+1} = y_i w\ell_i z_i \text{ or } x_i = y_i w\ell_i z_i \& x_{i+1} = y_i u\ell_i z_i.\]

This problem can be reduced from the halting problem of TM (Exercise).

Equational term rewriting systems (ETRS)

ETRS $R = (E, R)$
\[
E : \quad \text{equational theory } (F, E)
\]
$R : \quad \text{finite set of equations over } F, \text{ called rewrite rules}$
\text{(equation } s \approx t \text{ in } R \text{ is denoted by } s \rightarrow t)$
\[
\rightarrow_R : \quad \text{rewrite relation over } T_{F, V} \text{ defined as } s \rightarrow_R t \text{ if } \exists l \rightarrow r \in R : s \equiv_E C[l\theta] \& t \equiv_E C[r\theta] \text{ for some } C[], \theta
\]
\[
\rightarrow_R^* : \quad \text{the smallest reflexive and transitive relation containing } \rightarrow_R
\]
$=R : \quad \text{the smallest equivalence relation containing } \rightarrow_R$

Note

If $E = (F, \emptyset)$, we call $(E, R)$ a term rewriting system (TRS). For convenience, $((F, \emptyset), R)$ can be denoted by $(F, R)$. 


Confluence & termination

Let $\mathcal{R} : \text{ETRS over } F$

$\mathcal{R}$ is confluent if $\forall s t u : s \rightarrow^*_R t \& s \rightarrow^*_R u \Rightarrow \exists v : t \rightarrow^*_R v \& u \rightarrow^*_R v$

$\mathcal{R}$ is locally confluent if $\forall s t u : s \rightarrow_R t \& s \rightarrow_R u \Rightarrow \exists v : t \rightarrow^*_R v \& u \rightarrow^*_R v$

$\mathcal{R}$ is Church-Rosser if $\forall s t : s =_R t \Rightarrow \exists u : s \rightarrow^*_R u \& t \rightarrow^*_R u$

$\mathcal{R}$ is terminating if $\mathcal{R}$ does not admit any infinite sequence $t_1, t_2, t_3, \cdots$ such that $t_i \rightarrow^*_R t_{i+1} (i \geq 1)$

Note

$\mathcal{R}$ is confluent if and only if it is Church-Rosser

(The “if” is obvious by definition. The “only if” can be shown by the induction on the number of peaks $\rightarrow^*_R \cdots \rightarrow^*_R$ in $=_R$.)

Newman’s lemma

Every locally confluent and terminating (E)TRS is confluent.

Proof

Suppose $\mathcal{R} = (\mathcal{E}, \mathcal{R})$ is locally confluent and terminating. Define the binary relation $\triangleright = \rightarrow^*_R$. By definition, $\triangleright$ is a preorder. Moreover, since $\mathcal{R}$ is terminating, $\triangleright$ is well-founded, because $\triangleright - (\triangleright)^{-1}$ is the transitive closure of $\rightarrow_R$. Using the induction on trees with $\triangleright$, one can show that $\mathcal{R}$ is confluent: The base case is obvious, because $s \equiv t \equiv u$. For induction step, suppose $s \rightarrow^*_R t$ and $s \rightarrow^*_R u$. If $s \not\equiv t$ and $s \not\equiv u$ (otherwise, it is obvious), the right figure holds. We note that $s \triangleright w$ if $u' \triangleright w$ or $t' \triangleright w$, because by assumption, $s \triangleright t'$ and $s \triangleright u'$.

Critical pairs & local confluence

Suppose $l_1 \rightarrow r_1, l_2 \rightarrow r_2 \in \mathcal{R}$ are rewrite rules such that there exists a most general unifier $\theta$ for unifying $l_1$ and a sub-tree $l$ of $l_2$ (so $l_1\theta \equiv l\theta$). Let $l_2 = C[l]$, then $\langle C[r_1\theta], r_2\theta \rangle$ is called critical pair. It is known that a TRS $\mathcal{R}$ is local confluent if and only if its critical pairs are joinable ($\exists t : C[r_1\theta] \rightarrow^*_R t \& r_2\theta \rightarrow^*_R t$).
Knuth-Bendix procedure

Let $E : \text{set of equations}$
\[ \succ : \text{well-founded preorder} \& \text{closed under contexts and substitutions} \]

Given an equational theory $E = (F, E)$ with $\succ$, this procedure attempts to construct a terminating and confluent TRS $\mathcal{R} = (F, R)$ with $=_{\mathcal{R}} = =_{\mathcal{E}}$. When the procedure successfully terminates, it returns $R$ (set of rewrite rules); otherwise, it fails or runs forever. The procedure normalize is non-deterministic function that returns one of the candidates. Termination of $\mathcal{R}$ is a consequence of the compatibility with $\succ$. Confluence of $\mathcal{R}$ follows from the previous observation about critical pairs and Newman’s lemma.

**Procedure main($E, \succ$)**

if $\exists l \approx r \in E: l \not\succ r \& r \not\succ l$ then break(“fail”);
$R_0 := \{l \rightarrow r \mid l \approx r \text{ or } r \approx l \in E \text{ such that } l \succ r\}$;
while $\exists$ critical pair $(C[r_1\theta], r_2\theta)$ in $R_i$ is not joinable do
\[ s := \text{normalize}(R_i, C[r_1\theta]); \]
\[ t := \text{normalize}(R_i, r_2); \]
if $s \succ t$ then $R_{i+1} := R_i \cup \{s \rightarrow t\}$;
if $t \succ s$ then $R_{i+1} := R_i \cup \{t \rightarrow s\}$;
else break(“fail”);
\[ i++; \]
end; return $R_i$

**Procedure normalize($R, t$)**

if $t \not\rightarrow_{R}^* s \& s$ is not rewritten by $R$ anymore
then return $s$

---

**Proposition** [Knuth & Bendix]

If Knuth-Bendix procedure successfully terminates at the $n$-th loop, the TRS $\mathcal{R} = (F, R_n)$ is terminating and confluent with $=_{\mathcal{R}} = =_{\mathcal{E}}$.

**Proof**

Termination and confluence of $\mathcal{R}$ are verified in the previous page. Let $\mathcal{R}_i = (F, R_i)$ where $R_i$ is the set of rewrite rules obtained at the $i$-th loop, then we show that $=_{\mathcal{R}_i} = =_{\mathcal{E}}$ : Use the induction on the number of loops. The base case is obvious, because according to the procedure, every equation in $E$ is oriented by $\succ$, and thus $=_{\mathcal{R}_0} = =_{\mathcal{E}}$. For induction step, suppose $=_{\mathcal{R}_i} = =_{\mathcal{E}}$. If there is no more critical pair in $R_i$, the computation is done, so $\mathcal{R}_i = \mathcal{R}_{i+1}$. If there is a critical pair $(C[r_1\theta], r_2\theta)$, let $s = \text{normalize}(C[r_1\theta])$ and $t = \text{normalize}(r_2\theta)$, then $s =_{\mathcal{R}_i} C[r_1\theta]$ and $t =_{\mathcal{R}_i} r_2\theta$. This implies $s \approx t$, because $C[r_1\theta] =_{\mathcal{R}_i} r_2\theta$. Thus, regardless of the orientation of the equation $s \approx t$, we have $=_{\mathcal{R}_i} = =_{\mathcal{R}_i} \cup \{s \rightarrow t\}$ and $=_{\mathcal{R}_i} = =_{\mathcal{R}_i} \cup \{t \rightarrow s\}$. Hence, $=_{\mathcal{R}_i} = =_{\mathcal{R}_{i+1}}$, and therefore, by induction hypothesis, $=_{\mathcal{R}_{i+1}} = =_{\mathcal{E}}$. □

As a corollary of this proof, the correctness of the procedure can be preserved, even if the procedure is modified as follows : $R_{i+1} := \{ l \rightarrow r' \mid l \rightarrow r \in R_i \& r' = \text{normalize}(R_{i+1}, r) \} \cup \{ s \rightarrow t \}$ if $s \succ t$ (t → s if t ∼ s).

**Confluence modulo**

Let \((F, E)\) : equational theory \(E\)

\((F, R)\) : TRS \(R\)

\(R\) is confluent modulo \(E\)

\(\forall s_1, s_2, t_1, t_2 : s_1 \equiv_E s_2 \& s_1 \rightarrow^* R t_1 \& s_2 \rightarrow^* R t_2 \Rightarrow \exists u_1, u_2 : u_1 \equiv_E u_2 \& t_1 \rightarrow^* R u_1 \& t_2 \rightarrow^* R u_2\)

**Proposition (Huet 1980)**

1. If TRS \(R\) is terminating and \(\leftarrow_R \cdot \rightarrow_R \subseteq \rightarrow^*_R \cdot =_E \cdot \leftarrow^*_R\) (called locally confluent modulo \(E\)) and \(\leftarrow_R \cdot =_E \subseteq \rightarrow^*_R \cdot =_E \cdot \leftarrow^*_R\) (called locally coherent with \(E\)), then \(R\) is confluent modulo \(E\).

2. If ETRS \((E, R)\) is terminating and locally confluent modulo \(E\) and \(\leftarrow_R \cdot =_E \subseteq \rightarrow^*_R \cdot =_E \cdot \leftarrow^*_R\) (called locally coherent with \(\leftarrow_E\)), then \(R\) is confluent modulo \(E\).

**Exercise**

1. Let \(\succ = \{(a, b), (b, c), (c, a)\}\), then show that \(\succ\) is not well-founded. Show, on the other hand, that the transitive closure of \(\succ\) (the smallest transitive relation containing \(\succ\)) is well-founded.

2. Given a set \(S\) equipped with a well-founded preorder \(\succ\), show that \(S = \bigcup_{a \in \min_{\succ}(S)} \{x \in S \mid x \succ a\}\).

3. Show that the \(E\)-equivalence problem (page 9) is decidable for the classes of \(E_I, E_U, E_{ACI}, E_{ACU}, E_{ACIU}\).

4. Recall the word problem, which is the question if given words \(u, w\) are equivalent under given axioms \(u_i \approx w_i (u_i, w_i \in \Sigma^+, 1 \leq i \leq n)\). Show that the word problem is undecidable. (See page 10)

5. Complete by Knuth-Bendix procedure the equational theory \(E_{AIU}\).

For defining a strict order (the strict part of a well-founded preorder), use, e.g. the polynomial interpretation \(f(x, y) = 2x + y + 1\) and \(e = 1\) over positive integers.

6. Let \(E = (F, E)\). Show that if TRS \((F, R)\) is confluent modulo \(E\), then ETRS \((E, R)\) is confluent.
Appendix : Dauchet’s construction

It is undecidable for a given LBA (see page 18, seminar talk 2) whether it halts on any input [Caron 1991]

For each transition rule \( \langle p, a \rangle \rightarrow \langle q, b, X \rangle \) of LBA \( \mathcal{M} \), define a rewrite rule \( s \rightarrow t \) whose left-hand side \( s \) is

\[
f(g(x_1, a), p, h(x_2, x_3))
\]

and right-hand side \( t \) is

\[
f(g(g(x_1, b), x_2), q, x_3) \text{ if } X = R ; \ f(x_1, q, h(b, h(c, x_3))) \text{ if } X = L.
\]

E.g. for the input \( a b c \), if LBA \( \mathcal{M} \) selects the rule \( \langle q_0, a \rangle \rightarrow \langle p, d, R \rangle \), then one can have the reduction from \( f(g(\#_L, a), q_0, h(c, \#_R)) \) to \( f(g(g(\#_L, d), b), p, h(c, \#_R)) \). For the TRS \( \mathcal{R}_M \) associated to \( \mathcal{M} \), one can show that it is terminating if and only if \( \mathcal{M} \) halts on any input. However, since the latter problem is undecidable, termination problem for a (sub-)class of TRS is undecidable.

V. Turing Machine
Turing Machine (TM)

(Deterministic) Turing machine : \((\Gamma, \Sigma, Q, q_0, Q_{\text{fin}}, \Delta)\)

- **\(\Gamma\)** : alphabet containing special symbol \(\#\) (called *blank*)
- **\(\Sigma\)** : input tape symbols such that \(\Sigma \subseteq \Gamma - \{\#\}\)
- **\(Q\)** : finite set of *state symbols*
- **\(q_0\)** : start symbol such that \(q_0 \in Q\)
- **\(Q_{\text{fin}}\)** : final states such that \(Q_{\text{fin}} \subseteq Q\)
- **\(\Delta\)** : transition function, which is a partial function such that
  \[
  \langle p, a \rangle \mapsto \langle q, b, x \rangle \quad (p, q \in Q, a, b \in \Gamma, x \in \{L, R\})
  \]
  where, e.g. \(\langle p, a \rangle \mapsto \langle q, b, L \rangle\) means “a state \(p\) is changed to \(q\) after reading a character \(a\) and rewriting it to \(b\), and then the tape-head moves on the tape to the left.” If \(\langle q, b, R \rangle\) on the right-hand side in the above assignment, the tape-head moves to the right.

Turing Machine (cont’d)

Every TM \((\Gamma, \Sigma, Q, q_0, Q_{\text{fin}}, \Delta)\) is equipped with a single input tape which is divided into infinitely many *cells* such that a character in \(\Gamma\) can be stored in each cell.

TM has a single *tape-head* which reads a character in the cell, rewrites the character to some character, indicates the current state from \(Q\), and move the head on the tape to the left or right, according to \(\Delta\).

Initially, the input tape contains finite sequence of characters from \(\Sigma\) in the cells, and the remaining cells contains blanks (\(\#\)).

Besides, the tape-head indicates start symbol \(q_0\) and is located at the leftmost cell which contains a character. See below for example.

- initial configuration :

  \[
  \begin{array}{cccccccc}
  \ldots & \# & \# & a & \ldots & b & c & d & \ldots & e & \# & \# & \ldots \\
  \hline
  \end{array}
  \]
Infinity

Countable infinity ($\aleph_0$) : $\exists$ one-to-one correspondence with the list of natural numbers 1, 2, 3, ...

Georg Cantor (1845–1918)

... you turn up at the check-in counter of the Hotel Infinity only to find that the infinite number of rooms (numbered 1, 2, 3, 4, ... and so on, forever) are all occupied. The receptionist is perplexed — the Hotel is full — but the manager is unperturbed. No problem, he says: move the guest in room 1 to room 2, the guest in room 2 to room 3, and so on, forever. This leaves room 1 vacant for you to take and everyone still has a room!

(“Welcome to the Hotel Infinity”, chap. 3, by J.D. Barrow)

Note

Infinite hierarchy of infinity : $\aleph_0 \subseteq \aleph_1 \subseteq \aleph_2 \subseteq \aleph_3 \subseteq ...$

Accepted input tape

TM operates move by move, according to the transition function $\Delta$.

The machine either operates forever, or it halts when the tape-head with state $p$ reads character $a$ but there is no transition that matches $\langle p, a \rangle$.

For input-tape with a finite sequence $w$ of characters, if $\text{TM } M$ halts with final state $p$ in $Q_{\text{fin}}$, we say $w$ is accepted by $M$. The set of words accepted by $M$ is denoted by $L(M)$.

A language $L$ is called recursively enumerable if there exists $\text{TM } M$ that halts on each word in $L$, and it never halts on any word in $\Sigma^* - L$.

(In this definition, the set $Q_{\text{fin}}$ of final states does not play an essential role to define this class of languages)
Multi-tape TM (MTM)

Multi-tape TM is TM with \( n \) tapes (\( n \geq 1 \)) and \( n \) tape-heads (one head for each tape) but one state indicator.

Initially, the first tape contains the input (that is input symbols in the finite sequence of cells and blanks in the remaining cells), and the remaining tapes contain blanks.

And tape-head on the first tape is located at the leftmost of the input, and the other heads can be placed at arbitrary cells.

Transition function \( \Delta \) is the mapping:
\[
\langle p, a_1, \ldots, a_n \rangle \mapsto \langle q, b_1, \ldots, b_n, X_1, \ldots, X_n \rangle
\]
so that one transition admits simultaneous move of the multiple tape-heads.

2-tape TM

\[
\begin{array}{ccccccc}
\cdots & f & g & h & i & j & \cdots \\
\cdots & a & b & c & d & e & \cdots
\end{array}
\]

\[
\langle p, b, i \rangle \mapsto \langle q, k, \ell, R, L \rangle
\]

\[
\begin{array}{ccccccc}
\cdots & f & g & h & \ell & j & \cdots \\
\cdots & a & k & c & d & e & \cdots
\end{array}
\]

TM = MTM

Every language accepted by MTM is recursively enumerable

Proof

Let \( M \) be MTM with \( n \)-tapes whose set of tape symbols is \( \Gamma \). Suppose \( \diamond \) is a fresh symbol. Then, define a single tape TM \( M_S \) whose tape symbols are \( \Gamma \cup \{ \diamond \} \cup \{ \overline{a} \mid a \in \Gamma \} \). Consider \( n = 3 \) below, but the proof can be generalized for every \( n \geq 2 \):

1. For the input \( a_1 a_2 \cdots a_k \), \( M_S \) creates the word
\[
\diamond \overline{a}_1 a_2 \cdots a_k \diamond \overline{\diamond} \overline{\diamond},
\]
meaning that 1st head is placed on the leftmost character \( a_1 \), and the other heads are placed on the blank and cells on those tapes do not contain any character.

2. Simultaneous move on 3 tapes is simulated by the single tape that contains three finite segments (separated by \( \diamond \)). When one of heads rewrites \( a \) to \( b \) and moves to the right cell with \( c \), the corresponding word \( \overline{a} c \) on the single tape is rewritten to \( b \overline{c} \). If \( c = \diamond \), shift \( c \) and all symbols after \( c \) one cell to the right, and then rewrite \( \overline{\pi} \diamond \) to \( b \overline{\pi} \diamond \). Similar for the left. \( \square \)
Non-deterministic TM (NTM)

Non-deterministic TM $M = (\Gamma, \Sigma, Q, q_0, Q_{\text{fin}}, \Delta)$ where

$\Delta : \text{finite set of transition rules in the form of}$

$$\langle p, a \rangle \rightarrow \langle q, b, x \rangle$$

$p, q \in Q, a, b \in \Gamma, x \in \{ L, R \}$

$M$ accepts word $w$ if and only if there exists a sequence of moves of the tape head that leads from the initial configuration to the situation where the head halts with a final state (a final configuration).

**Proposition**

$\text{TM} \subseteq \text{NTM}$

**Proof**

Because every TM is NTM, languages accepted by TM are accepted by NTM.

$\text{TM} = \text{NTM}$

Every language accepted by NTM is recursive enumerable

**Proof**

For NTM $M$ with $\Gamma$, define 3-tape TM $M_3$ with $\Gamma \cup \{ \diamond, 0, 1 \}$: Tape 1 is a simulation tape, which means this tape contains a configuration (a copy of tape) on $M$’s computation. Tape 2 contains nodes of $M$’s computation tree which are already or will be visited by simulation. Nodes on tape 2 are separated by $\diamond$. Tape 3 contains the next address or the empty word (which means no address left to visit for next). The simulation proceeds in breadth-first manner. If $M$ accepts the input on tape 1, $M_3$ enter a final state. If $M$ halts on a non-final state, $M_3$ traverses $M$’s computation tree. If $M$ meets a non-deterministic choice in the current computation on tape 1, keep in tape 2 this address (denoted by a word in $\{0,1\}^*$) and the current configuration, and then follow the way of backtracking computation. If $M$ halts on a non-final state and tape 3 is the empty word, $M_3$ halts on a non-final state.
Recursive languages

Given an alphabet $\Sigma$

A language $L$ over $\Sigma$ is **recursive** if there exists a TM $\mathcal{M}$ such that $L(\mathcal{M}) = L$ and $\mathcal{M}$ halts on any input in $\Sigma^*$ ($\mathcal{M}$ is called **halting** TM).

**Note 1**

Let $\mathcal{M}$ be halting TM $(\Gamma, \Sigma, Q, q_0, Q_{\text{fin}}, \Delta)$, then $\mathcal{M}_c = (\Gamma, \Sigma, Q, q_0, Q - Q_{\text{fin}}, \Delta)$ accepts the complement of $L(\mathcal{M})$, because TM is deterministic. Therefore, recursive languages are closed under complement. Since recursive languages are also closed under union, by de Morgan’s law, this class is closed under Boolean operations.

**Note 2**

One can transform halting TM $\mathcal{M}$ to $\mathcal{M}'$ such that for input $w$, $\mathcal{M}'$ halts if $\mathcal{M}$ halts with a state in $Q_{\text{fin}}$; otherwise (if $\mathcal{M}$ halts with a state in $Q - Q_{\text{fin}}$), $\mathcal{M}'$ does not halt. Hence, recursive languages are recursive enumerable.

**Question**

Are recursive enumerable languages recursive?

---

**Binary encoding**

$C(\text{TM})$ : set of TM’s for languages over $\{0, 1\}$ defined as follows

$\mathcal{M} \in C(\text{TM})$ if $\mathcal{M} = (\{0, 1, \#\}, \{0, 1\}, Q, 10, Q_{\text{fin}}, \Delta)$ where $Q$ is a finite subset of $\{1 \cdot 0^n | n \geq 1\}$ and $Q_{\text{fin}}$ contains 100 only.

$\Psi :$ mapping from $\Delta$ to $\{0, 1\}^*$ such that

$$\Psi(\langle p, a \rangle \mapsto \langle q, b, X \rangle) = p \psi(a) q \psi(b) \psi(X)$$

where

$$\psi(x) = \begin{cases} 
10 & \text{if } x = 0 \text{ or } x = L \\
100 & \text{if } x = 1 \text{ or } x = R \\
1000 & \text{if } x = \# 
\end{cases}$$

E.g. $\Psi(\langle 10, \# \rangle \mapsto \langle 100, 0, L \rangle) = 10 1000 100 100 10$
Let $\mathcal{M} = (\{0, 1, \#\}, \{0, 1\}, Q, 10, Q_{\text{fin}}, \Delta)$ with the mappings in $\Delta$:

$$\langle p_1, a_1 \rangle \mapsto \langle q_1, b_1, X_1 \rangle \ldots \langle p_n, a_n \rangle \mapsto \langle q_n, b_n, X_n \rangle$$

Define the word for $\mathcal{M}$:

$$\Psi(\langle p_1, a_1 \rangle \mapsto \langle q_1, b_1, X_1 \rangle)110 \cdots 110 \Psi(\langle p_n, a_n \rangle \mapsto \langle q_n, b_n, X_n \rangle)$$

**Note**

Let $w$ be the above word,

- $w$ can be defined in $|\Delta|!$ different ways (one-to-many mapping)
- $w$ does not contain any $1^n$ for $n \geq 3$

Moreover,

- TM $\mathcal{M}$ can be retrieved from $w$

**Diagonalization language**

Let $\mathcal{M}_\varnothing = (\{0, 1, \#\}, \{0, 1\}, \{10\}, 10, \varnothing, \varnothing)$

Define

$$L_{DL} = \{ w \in \{0, 1\}^* \mid w \text{ is not accepted by } \text{decode}(w) \}$$

where $\text{decode}(w) = \mathcal{M}$ if $w$ represents TM $\mathcal{M}$; otherwise, $\mathcal{M}_\varnothing$

**Proposition**

No TM accepts $L_{DL}$

**Proof**

Suppose, by way of contradiction, that TM $\mathcal{M}$ accepts $L_{DL}$. One can modify $\mathcal{M}$ to $\mathcal{M}'$ whose initial state is 10 and final state is 100 only. And all state symbols of $\mathcal{M}'$ are renamed with symbols from $\{10^n \mid n \geq 1\}$. Then $\mathcal{M}'$ can be translated by $\Psi$ to a word $w$ over $\{0, 1\}$. If $w \in L_{DL}$, $w$ is accepted by $\mathcal{M}'$, but then $w \notin L_{DL}$. If $w \notin L_{DL}$, that means $w$ is not accepted by $\mathcal{M}'$, then $w \in L_{DL}$. We conclude by this contradiction, that $\mathcal{M}'$ does not exist, and hence, $\mathcal{M}$ does not exist either.
**Complement of $L_{DL}$**

Take the complement of $L_{DL}$, that is,

$$(L_{DL})^c = \{ w \in \{0, 1\}^* \mid w \text{ is accepted by } \text{decode}(w) \}$$

**Proposition**

There exists TM that accepts $(L_{DL})^c$

**Proof (sketch)**

Define TM $M$ such that (1) for input $w$, first $M$ duplicates it as the pair of word $w$ and TM $M_w (= w)$, and (2) verifies whether $M_w$ is TM defined by binary encoding. (3) If yes, $M$ starts to simulate the move of $M_w$. (4) When $M_w$ enters the final state and halts, $M$ accepts $w$; when $M_w$ halts with a non-final state, $M$ also halts without accepting $w$. ☐

**Corollary**

No halting TM accepts $(L_{DL})^c$

---

**Universal TM (UTM)**

There exists TM $M_U$ that simulates on input $\langle x, y \rangle$, TM $\text{decode}(x)$ on input $y$ such that $L(M_U) = \{ \langle x, y \rangle \mid y \in L(\text{decode}(x)) \}$

**Proof**

Similar to the previous proof, but to be precise, first define 3-tape TM $M$. For the input $\langle x, y \rangle$, $x$ is copied to tape 2, and verify whether $x$ is (deterministic, single-tape) TM defined by binary encoding. If yes, $M$ simulates $x$’s computation on tape 1. The current state of $x$ and other information needed for simulation are stored in tape 3. Because the above MTM $M$ can be simulated by a single-tape TM, take that TM for $M_U$. ☐

The smallest UTM over $\{0, 1\}$ known so far needs 22 state symbols [1]. Other small UTM’s, in order of $|Q|$ (the number of state symbols) $\times |\Gamma - \{$\#$$\}$ (the number of tape symbols), known so far are: $2 \times 18$, $3 \times 10$, $4 \times 6$, $5 \times 5$, $7 \times 4$, $10 \times 3$ [2].


Computability and Undecidability

Where does undecidability come from?

**Proposition**

The set of functions from \( \{0,1\}^* \) to \( \{0,1\} \) is not countable

**Proof**

Suppose, by way of contradiction, that the above set is represented as \( \{f_i \mid i \geq 0\} \). Let \( a_0, a_1, \ldots, a_i, \ldots \) be a sequence of words over \( \{0,1\} \) in the lexicographic order. Define a function \( f \) from \( \{0,1\}^* \) to \( \{0,1\} \) such that for each \( a_i, f(a_i) = 1 \) if \( f_i(a_i) = 0 \); \( f(a_i) = 0 \) if \( f_i(a_i) = 1 \). Then, \( f \) is different from any \( f_i \) \((i \geq 0)\), leading to the contradiction.

**Definition**

A function \( f \) from \( \Sigma^* \) to \( \{0,1\} \) is **computable** if there is a halting TM \( M \) such that \( w \in L(M) \) if \( f(w) = 1 \); \( w \notin L(M) \) if \( f(w) = 0 \)

Only proper subset of functions from \( \{0,1\}^* \) to \( \{0,1\} \) is computable

**Rice’s theorem**

Every non-trivial property of recursively enumerable languages is undecidable

A **property** of recursively enumerable languages is a subset of recursively enumerable languages. A property is **non-trivial** if it is not the empty set and not the set of all recursively enumerable languages. The above theorem states that for every non-trivial property \( P \), the question if, for a given TM \( M \), \( L(M) \in P \) is undecidable.

**Proof**

Suppose, by way of contradiction, that \( P \) is decidable, i.e. that there is a halting TM \( M_1 \) accepting binary codes of TM’s that accept languages in \( P \). If \( \emptyset \notin P \), let \( M_L \) be TM accepting a non-empty language \( L \) in \( P \). There must be such \( L \) in \( P \), since \( P \) is non-trivial. Next, define TM \( M_2 \) such that, depending on a given TM \( M \) and input \( w \), \( M_2 \) accepts \( L \) or \( \emptyset \): If \( M \) does not accept \( w \), \( M_2 \) does not accept any input, so \( M_1 \) does not accept binary code of \( M_2 \). If \( M \) accepts \( w \), \( M_2 \) simulates TM \( M_L \), so \( M_1 \) accepts binary code of \( M_2 \). Then, \( M \) accepts \( w \) if and only if \( M_1 \) accepts binary code of \( M_2 \). If \( \emptyset \in P \), take the complement of \( P \), and then apply the above argument. Because \( P \) is a recursive language, \( P^C \) is also decidable. However, our assumption leads to the contradiction, since the halting problem whether \( M \) accepts \( w \) is undecidable. Hence, \( P \) is not decidable.
Trakhtenbrot’s theorem

The problem deciding if a first-order sentence is \textit{finitely satisfiable} (i.e. has a finite model) is undecidable.

**Corollary 1**

The set of \textit{finitely valid} sentences (i.e. valid sentences, each of which has a finite model) is \textit{not} recursively enumerable.

**Proof of Corollary 1**

It is not difficult to see that the set of finitely satisfiable sentences (i.e. satisfiable sentences, each of which has a finite model) is recursively enumerable. Now we suppose, for leading to a contradiction, that the set of finitely valid sentences, denoted $S$, is also recursively enumerable. Since a sentence $\phi \in S$ is finitely valid iff $\neg \phi$ is not finitely satisfiable, $S$ and $(S)^c$ are recursive, because $(S)^c$ is recursively enumerable and $S$ is so by assumption. Note that $(S)^c$ is the set of finitely satisfiable sentences. However, it contradicts to the above Trakhtenbrot’s undecidability theorem. \hfill \Box

**Corollary 2**

An effective axiomatization of first-order logic over finite models is \textit{not} possible.

**Proof of Trakhtenbrot’s theorem**

We encode a TM $M$ to a first-order sentence $\Psi_M$ such that: $M$ halts on the empty input iff $\Psi_M$ is finitely satisfiable. Let $M = (\Gamma, \Sigma, Q, q_0, Q_{\text{fin}}, \Delta)$ where $\Gamma = \{0, 1, \#\}$. We take the set $\sigma$ of function symbols $(0, s, p)$ and predicate symbols $\text{T}_c$, $\text{H}_q$, $\text{T}_c^{\text{H}_q}$ for all $q \in Q$.

Here $\text{T}_c(p, t)$ means a cell at position $p$ at time $t$ contains a character $c$ ($c \in \{0, 1, \#\}$); $\text{H}_q(p, t)$ means the tape-head is located at position $p$ and in state $q$ at time $t$. The predicate $\prec$ is a linear order of $\cdots \prec p(p(0)) \prec p(0) \prec s(0) \prec s(s(0)) \prec \cdots$, which is definable by a first-order sentence, denoted $\Psi_{\prec}$. Define eight other first-order sentences as follows:

- $\Psi_{\text{tape}} : \forall p, t \left[ \bigwedge_{c \in \Gamma} (\text{T}_c(p, t) \leftrightarrow \bigwedge_{d \in \Gamma - \{c\}} \neg \text{T}_d(p, t)) \right]$
- $\Psi_{\text{head}} : \forall t \exists p \forall q \in Q \text{H}_q(p, t) \land \forall t, p_1, p_2, q_1, q_2 (\text{H}_{q_1}(p_1, t) \land \text{H}_{q_2}(p_2, t) \Rightarrow p_1 = p_2 \land q_1 = q_2)$
- $\Psi_{\text{init}} : \text{H}_{q_0}(0, 0) \land \forall p \text{T}^p_0(p, 0)$
- $\Psi_{\text{tran-r1}} : \bigwedge_{(q_1, c_1) \rightarrow (q_2, c_2, R)} \left[ \forall p, t \left\{ \left( 0 \prec p \lor p = 0 \right) \land \text{H}_{q_1}(p_1, t) \land \text{T}_{c_1}(p_1, t) \Rightarrow \text{H}_{q_2}(s(p), s(t)) \land \text{T}_{c_2}(p, s(t)) \land \forall r \left( p \neq r \Rightarrow \bigwedge_{c=0,1,\#} \text{T}_c(r, t) \leftrightarrow \text{T}_c(r, s(t)) \right) \right\} \right]$

(Proof cont’d)
Let $\Psi_M$ be the conjunction of all the previously defined sentences. By construction, $\Psi_M$ has a finite model $A$ iff $A$ represents a computation of $M$, which $M$ halts on the empty input. Since the halting problem of TM even with the empty input is undecidable (Rice’s theorem), so is the finite satisfiability.

Exercise

1. Show that language $L$ is recursively enumerable if and only if there exists TM which accepts $L$.
2. Show that languages accepted by non-deterministic halting TM are recursive.
3. Show that $L(M_U)$ is recursively enumerable, but it is not recursive. (Cf. Halting problem)
4. Show that halting TM’s are strictly more powerful than LBA.
   E.g., use binary encoding. Suppose $\xi_L, \xi_R \in \Sigma$, each of which is left-end and right-end of LBA, respectively. One can construct a halting TM that accepts binary codes of LBA, because a given code of non-deterministic TM is LBA if and only if $\Delta$ does not contain a transition rule $\langle p, p_L \rangle \rightarrow \langle q, a, L \rangle$ or $\langle p, p_R \rangle \rightarrow \langle q, a, R \rangle$ for some states $p, q \in Q$ and tape symbol $a$. Then, define the language $L = \{ w \in \{0,1\}^* | w$ is not accepted by decode($w$) $\}$. Show that $L$ is recursive, and $L$ is not accepted by any LBA.
5. Is the property $P = \{\emptyset\}$ of recursively enumerable languages decidable or undecidable? Explain the reason why.
Counter machine [1; also Lambek, Melzak 1961] consists of

\[ R \]: finite set of registers \( r_0, \ldots, r_n \) \((n \geq 0)\), each of which can store any non-negative integer

\[ \Delta \]: finite set of labeled instructions with the following forms

\[ \ell_i, \text{INC}(r_i) \]: Increment \([r_i]\) by 1 and go to \(\ell_{i+1}\)

\[ \ell_i, \text{DEC}(r_i) \]: Decrement \([r_i]\) by 1 if \(r_i\) is not 0, and go to \(\ell_{i+1}\)

\[ \ell_i, \text{JZ}(r_i, \ell_k) \]: Jump to the instruction labeled \(\ell_k\) if \([r_i]\) equals 0, otherwise go to \(\ell_{i+1}\)

where \([r_i]\) means the content of register \(r_i\)

Initially, non-negative integers as input are stored in registers, and the machine executes first the instruction labelled \(\ell_0\). The machine halts if the next instruction does not exit, and then the input is accepted.

The class of 2-counter machines is as expressive as TM’s.

VI. P & NP
**TM & k-TM**

\( k\)-TM \((k \geq 1)\) : MTM with

- one control-unit
- one read-only tape (called input tape)
- \(k\) read/write tapes (called working tapes)

\(k\)-NTM : non-deterministic \(k\)-TM

Cf. TM \(\triangleq\) one control-unit \(\varnothing\) + one read/write tape

**Corollary**

\[ \text{TM} = k\text{-TM} = k\text{-NTM} \ (k \geq 1) \]

**Proof**

The left equivalence follows from the fact that MTM is simulated by TM and the reverse is trivial. Using a similar proof of the previous fact (see the slides of the previous talk), one can show that \(k\)-NTM is simulated by NTM and the reverse also holds. Since NTM is simulated by TM, the right equivalence follows.

---

**Running time & working space**

Given \(k\)-TM \(M = (\Gamma, \Sigma, Q, q_0, Q_{\text{fin}}, \Delta)\)

\[ \text{time}_M(w) : \text{the number of moves until } M \text{ halts on input } w \]
\[ \text{(called running time of } M \text{ on } w) \]
\[ \text{space}_M(w) : \text{the number of cells in working tapes which } M \]
\[ \text{visited at least once until } M \text{ halts on input } w \]
\[ \text{(called working space of } M \text{ on } w) \]

For the length \(n\) of input

\[ T_M(n) : \max\{ \text{time}_M(w) \mid w \in \Sigma^* : |w| = n \} \]
\[ S_M(n) : \max\{ \text{space}_M(w) \mid w \in \Sigma^* : |w| = n \} \]

**Question**

How should we define \(T_L\) (or \(S_L\)) for language \(L\)?
Given $k$-TM $M_1$, there exists $(k+1)$-TM $M_2$ such that

- $\mathcal{L}(M_1) = \mathcal{L}(M_2)$
- $T_{M_2}(n) \leq \frac{1}{2} T_{M_1}(n)$ for sufficiently large $n \in \mathbb{N}$

if $\lim_{n \to \infty} T_{M_1}(n)/n = \infty$

**Proof**

Consider $k = 1$, but the following proof can be generalized to arbitrary $k \geq 1$.

Let $M_1 = (\Gamma, \Sigma, Q, q_0, Q_{\text{fin}}, \Delta)$, then define $M_2 = (\Gamma', \Sigma, Q', q'_0, Q'_{\text{fin}}, \Delta')$ with $\Gamma' = (\Gamma \times \Gamma \times \Gamma) \cup \Sigma$, $Q' = \{ (p, i) \mid p \in Q, 1 \leq i \leq 3 \} \cup \{ q'_0, q_{\text{acc}}, q_{\text{rej}} \}$ and $Q'_{\text{fin}} = \{ q_{\text{acc}} \}$.

Here the blank for $M_2$ is $(\#, \#, \#)$, and the states $q'_0, q_{\text{acc}}, q_{\text{rej}}$ are fresh symbols. Transition function $\Delta'$ is defined according to $M_1$’s move on each 3 cells (“chunk” of cells at $2i-1, 2i, 2i+1$). That is, initially, for input $a_0 a_1 \cdots a_n$, $M_2$ copies it to $(k+1)$th working tape in compressed manner, e.g. $(\#, a_0, a_1) (a_1, a_2, a_3) \cdots (a_{n-1}, a_n, \#)$ if $n$ is even; $(\#, a_0, a_1) (a_1, a_2, a_3) \cdots (a_{n-2}, a_{n-1}, a_n)$ if $n$ is odd. For this initialization, including the running time to return the tape-head on $(k+1)$th working tape to the leftmost position, $M_2$ needs $n + \frac{1}{2}n + c$ moves ($c$ constant). Next, using $(k+1)$th working tape as the input tape of $M_2$, $M_2$ simulates $M_1$. (proof cont’d)

**Proof (cont’d)**

For the move of $M_2$, suppose, for instance, that $M_1$ with the state $p$ reads $y_i$ on the input tape, and then $M_1$ moves on the working tape as shown in the figure. If $M_1$’s tape-head on the input tape stays, say on $y_{i+1}$, in the same chunk even after the other head moves on to the cell of $b_6$, $\Delta'$ contains the following map:

$\langle (p, 2), (\#, \#, \#), (y_{i-1}, y_i, y_{i+1}), (b_3, b_4, b_5) \rangle \mapsto \langle (q, 3), (\#, \#, \#), (y_{i-1}, y_i, y_{i+1}), (c_1, c_2, c_3), S, R, S \rangle$

On the right-hand side of the above mapping, “$\text{S}$” stands for stay. Because $M_2$’s head on the input tape is no longer needed, it keeps staying on a blank. The head on $(k+1)$th-working tape also stays on the same cell, but the the current state must be changed from $(p, 2)$, meaning that $M_1$’s tape-head reads $y_i$, to $(q, 3)$, meaning that the head reads $y_{i+1}$. During the computation, $M_2$ should halt on $q_{\text{acc}}$ (resp. $q_{\text{rej}}$) if $M_1$ accepts (rejects) the input. Moreover, since $M_1$ is deterministic, $M_2$ must be deterministic. By construction, the simulation can be done within at most $\frac{1}{2} T_{M_1}(n) + c'$ ($c'$ constant) moves. Since $\lim_{n \to \infty} T_{M_1}(n)/n = \infty$, we have that for large $n \in \mathbb{N}$, $T_{M_2}(n) \leq \frac{1}{2} T_{M_1}(n)$. □
Remarks

1. The size of chunk is not necessarily 3. When the size is \( i \geq 3 \), the simulation can be done within \( \frac{1}{i-1} T_{M_1}(n) \)

2. \( \mathcal{M}_2 \) requires \(|\Sigma|^i \) tape symbols and \((|Q|\times i) + 3\) state symbols

3. If \( \lim_{n\to\infty} \frac{T_{M_1}(n)}{n^2} = \infty \), one can construct \( k \)-TM \( \mathcal{M}_2 \) that simulates \( k \)-TM \( \mathcal{M}_1 \) (without introducing a new working tape)

4. From the proof, one can see that \( S_{M_2}(w) \) is also improved:

**Proposition (Tape compression)**

Given \( k \)-TM \( \mathcal{M}_1 \), there exists \((k+1)\)-TM \( \mathcal{M}_2 \) such that

- \( \mathcal{L}(\mathcal{M}_1) = \mathcal{L}(\mathcal{M}_2) \)
- \( S_{M_2}(n) \leq \frac{1}{2} S_{M_1}(n) \) for sufficiently large \( n \in \mathbb{N} \)

Note

If the size of the chunk is \( i \) \((i \geq 3)\), the simulation can be done within \( \frac{1}{i-1} S_{M_1}(n) \)

---

**DTIME & DSPACE**

Let \( T, S \) be 1-variable polynomials with positive coefficients

\( \text{DTIME}(T) \) : languages accepted by \( k \)-TM \( \mathcal{M} \) such that

\[ T_{\mathcal{M}}(n) \leq T(n) \text{ for every (sufficiently large) } n \in \mathbb{N} \]

\( \text{DSPACE}(S) \) : languages accepted by \( k \)-TM \( \mathcal{M} \) such that

\[ S_{\mathcal{M}}(n) \leq S(n) \text{ for every (sufficiently large) } n \in \mathbb{N} \]

Let \( \mathcal{P} \) be the set of 1-variable polynomials with positive coefficients

\[ \mathcal{P} : \bigcup_{T \in \mathcal{P}} \text{DTIME}(T) \]

\[ \text{PSPACE} : \bigcup_{S \in \mathcal{P}} \text{DSPACE}(S) \]

Note

\( \mathcal{P} \subseteq \text{PSPACE} \)

(\( \because \) Polynomially time-bounded computation requires at most polynomial tape space)
Let $T, S$ be 1-variable polynomials with positive coefficients

$\text{NTIME}(T)$ : languages accepted by $k$-NTM $M$ such that
$T_M(n) \leq T(n)$ for every (sufficiently large) $n \in \mathbb{N}$

$\text{NSPACE}(S)$ : languages accepted by $k$-NTM $M$ such that
$S_M(n) \leq S(n)$ for every (sufficiently large) $n \in \mathbb{N}$

Let $\mathcal{P}$ be the set of 1-variable polynomials with positive coefficients

$\text{NP} : \bigcup_{T \in \mathcal{P}} \text{NTIME}(T)$

$\text{NPSPACE} : \bigcup_{S \in \mathcal{P}} \text{NSPACE}(S)$

**Note**

$\mathcal{P} \subseteq \text{NP} \subseteq \text{NPSPACE}, \ \text{PSPACE} \subseteq \text{NPSPACE}$

**Example of P**

The membership problem for CFG in Chomsky normal form:

instance is grammar $\mathcal{G} = (\Sigma, Q, q_0, Q_{\text{fin}}, \Delta)$, word $w$ in $\Sigma^*$
solution is “yes” if $w \in L(\mathcal{G})$ ; “no” otherwise

**Proof (for the problem being in P)**

Let $w = a_1 a_2 \cdots a_n$ ($n \geq 0$). By assumption of the problem, transition rules in $\Delta$
are in the forms of $p \rightarrow qr, p \rightarrow a, q_0 \rightarrow \varepsilon$  for $p \in Q, q, r \in Q - \{q_0\}, a \in \Sigma$.

1. If $n = 0$ and $q_0 \rightarrow \varepsilon \in \Delta$, return “yes.”

2. Next, execute the following program:
   - for ($i = 1$ , $i \leq n$ , $i++$)
     - if $\exists p \rightarrow a_i$ in $\Delta$
       - $T(i,i) := T(i,i) \cup \{p\}$
   - for ($k = i$ , $k \leq j$ , $k++$)
     - if $\exists p \rightarrow qr$ in $\Delta$
       - $\exists q \in T(i,k)$ & $\exists r \in T(k+1,j)$
       - then $T(i,j) := T(i,j) \cup \{p\}$

The innermost loop is done in $k_1 \times n$ running time, hence this computation is done in
at most $k_2 \times n^3$ ($k_1, k_2$ constants). This implies that the problem is in $\text{DTIME}(n^3)$.
The bounded halting problem for NTM:

instance is $\text{NTM} \ M = (\Gamma, \Sigma, Q, q_0, Q_{\text{fin}}, \Delta)$, $w$ in $\Sigma^*$, $k$ in $\mathbb{N}$
solution is “yes” if $w \in L(M)$ within $k$ moves; “no” otherwise

Proof

Suppose $\langle \Psi(M), \Psi(w) \rangle$ is the binary code for non-deterministic UTM $M_{\text{NU}}$, then the question if $w \in L(M)$ is the membership problem $\langle \Psi(M), \Psi(w) \rangle \in L(M_{\text{NU}})$. Let $n$ be the size of input for this problem, then the total length of the word that represents $M$ and $w$ with delimiters is at most $2|\Gamma| + 2|Q| + |\Gamma|^2 \times |Q|^2 + |w| + c_1$. Encoding $M$ and $w$ by $\Psi$ takes $(c_2 \times |\Gamma|^2 \times |Q|^2) + (c_3 \times |w|)$ running time in proportion to $n$. The simulation of $M$ takes $(c_4 \times |\Gamma|^2 \times |Q|^2) + c_5$ for each move. In this simulation, at each step, $M_{\text{NU}}$ tries to find a transition rule which can be applied, and then reflects the result on the tapes. By assumption, the simulation must be stopped within $k$ iterations. Hence, the problem is in NTIME($n$).

Consult books (e.g. [1]) or search on the web for more examples of NP-problems.


Polynomial slow-down (simulation of MTM).

2-TM $M_1$ can be simulated by TM $M_2$ in $c \times T_{M_1}(n)^2$ ($c$ : constant)

Proof

Let $\Gamma$ be the set of tape symbols of $M_1$. Similar to the proof of “MTM=TM” in the previous talk on TM, suppose $\diamond$ is a fresh symbol, and then, define (single tape) TM $M_2$ whose tape symbols are $\Gamma \cup \{\diamond\} \cup \{\pi \mid a \in \Gamma\}$. In this proof, $M_2$’s single tape is divided into 2 tracks as shown in the figure. The symbols in input tape and working tape are alternately placed on the single tape. The left- and right-end of the sequence are marked by $\diamond$. The locations of $M_1$’s tape-heads on the input tape and on the working tape are represented by symbols with overline, e.g. $\overline{\pi}$. Let $n$ be the length of the input. For each move, the simulation takes at most $2 \times T_{M_1}(n) + 9$ moves, where $8 (= 2 \times 4)$ moves for adjusting the head and overwriting overlined-symbols to normal symbols, and 1 move for reading $\diamond$. Since the number of iterations is $T_{M_1}(n)$, the simulation takes $2 \times T_{M_1}(n)^2 + 9 \times T_{M_1}(n)$ in total. For the initialization, it takes $c_1 \times n^2$ ($c_1$ constant). $\square$

Note that the above proof can be generalized to the simulation of $k$-TM ($k \geq 2$).
Summary of TM & MTM & NTM

NTM is simulated by MTM in P-time iff P = NP

Reducibility

Given languages $L$ over $\Sigma$, $M$ over $\Gamma$ ($\Sigma \subseteq \Gamma$)

$L \leq_{m} M$ : $L$ is many-one reducible to $M$ if

$\exists k$-TM $M$ such that for every input $w \in \Sigma^*$,

$w \in L$ if and only if $M$ halts on $w$ with $w' \in M$

as output on working tape $k$

($\exists f$ computable function : $w \in L$ iff $f(w) \in M$)

$L \leq_{P} M$ : $L$ is polynomial-time reducible to $M$ if

$L \leq_{m} M \& T_M(n) \leq T(n)$ for polynomial $T$

($\exists f$ polynomial function : $w \in L$ iff $f(w) \in M$)

$L \leq_{log} M$ : $L$ is log-space reducible to $M$

$L \leq_{m} M \& S_M(n) \leq c \times \log n$ ($c$ : constant)

* $k$th-tape is write-only and is not taken into account as working space
**P- and NP-completeness**

Given language \( L \) over \( \Sigma \)

- \( L \) is called \textbf{NP-hard} if for every language \( M \in \text{NP} \), \( M \leq^P_m L \)
- \( L \) is called \textbf{NP-complete} if \( L \) is NP-hard \& \( L \in \text{NP} \)
- \( L \) is called \textbf{P-hard} if for every language \( M \in \text{P} \), \( M \leq^{\log}_m L \)
- \( L \) is called \textbf{P-complete} if \( L \) is P-hard \& \( L \in \text{P} \)

**Note**

\[ \leq^{\log}_m \subseteq \leq^P_m \]

\[ \therefore \text{DSPACE}(T(n)) \subseteq \text{NSPACE}(T(n)) \subseteq \bigcup_{c>0} \text{DTIME}(2^{cT(n)}) \text{ if } T(n) \geq \log(n) \]

Moreover, the following statements are equivalent:

1. \( P = \text{NP} \) \hspace{1cm} (1 \Rightarrow 3 \text{ From previous observation})
2. NP-complete problem is in P \hspace{1cm} (2 \Rightarrow 1 \text{ P,NP are closed under } \leq^P_m)
3. P-complete problem is NP-complete \hspace{1cm} (3 \Rightarrow 2 \text{ Obvious})

**P- and NP-complete problems (tricky example)**

The bounded halting problem for TM:

- instance is \( \text{TM } \mathcal{M} = (\Gamma, \Sigma, Q, q_0, Q_{\text{fin}}, \Delta) \), \( w \) in \( \Sigma^* \), \( k \) in \( \mathbb{N} \)
- solution is “yes” if \( w \in \mathcal{L}(\mathcal{M}) \) within \( k \) moves; “no” otherwise

This problem is P-complete

**Proof**

Similar to the bounded halting problem for NTM, it is easy to show that this problem is in P. Next, let \( L \) be a language over \( \Sigma \) in P. Then, there exists TM \( \mathcal{M}_L \) with polynomial \( T(n) \) such that \( L = \mathcal{L}(\mathcal{M}_L) \) and for every \( w \in L \), \( \mathcal{M}_L \) halts on \( w \) within \( T(|w|) \) running time. Take this TM \( \mathcal{M}_L \), an (arbitrary) word \( w \) from \( \Sigma^* \), and \( k = T(|w|) \), then \( w \in L \) if and only if \( \mathcal{M}_L \) halts on \( w \) within \( k \) moves.

The above example can be modified as an example of NP-complete problems, which means that:

The bounded halting problem for NTM is NP-complete
Other NP-complete problems

Satisfiability problem for Boolean formulas [Cook 1971]:
- instance is propositional Boolean formula $\phi$
- solution is “yes” if there is an assignment that satisfies $\phi$ ;
  “no” otherwise

Hamilton circuit in directed graphs:
- instance is directed graph $G$
- solution is “yes” if there is a cycle that passes through every
  vertex in $G$ exactly once ; “no” otherwise

Clique in graphs:
- instance is directed graph $G$, $k$ in $\mathbb{N}$
- solution is “yes” if $G$ has a complete sub-graph (each pair of
  vertices in the sub-graph is connected by an edge)
  of size $k$ ; “no” otherwise

Savitch’s theorem

PSPACE = NPSPACE

Proof

It suffices to show $\subseteq$, because the reverse is trivial. Let $L$ be a language in
PSPACE, then there exists $k$-NTM $\mathcal{M}_1 = (\Gamma, \Sigma, Q, q_0, Q_{\text{fin}}, \Delta)$ which halts on input
$w$ using at most $S(|w|)$ space. Suppose $k = 1$, but this proof can be generalized to
any $k \geq 1$. We construct below $k'$-TM $\mathcal{M}_2$ that accepts $L$ and that uses at most
$S(|w|)^2$ space. Assume for $\mathcal{M}_2$ that it accepts $w$ when and only when $\mathcal{M}_2$ clean up
the working tape (that contains only blanks) and halts with the final state $q_{\text{fin}}$ and
the tape-head on the input tape is placed at leftmost position.

Observation: Since $\mathcal{M}_1$ uses for input $w$ at most $S(|w|)$ space, $w \in L$ if and only if
$w$ is accepted within $|\Gamma|^{S(|w|)} \times (|Q| \times S(|w|) \times |w|)$ moves, where $|\Gamma|^{S(|w|)}$ is the number
of possible configurations of working tape and $(|Q| \times S(|w|) \times |w|)$ is the possible
combinations of state and positions of (two) tape-heads. Hence, $w \in L$ if and only if
$w$ is accepted within $2^{c \times S(|w|)}$ moves ($c$ : constant and computable).

This observation brings the idea to define $k'$-TM $\mathcal{M}_2$ such that $\mathcal{M}_2$ accepts input
$w$ if $w$ is accepted by $\mathcal{M}_1$ within $2^{c \times S(|w|)}$ moves ; $\mathcal{M}_2$ rejects $w$ otherwise. One
should notice that $\mathcal{M}_2$ does not necessarily simulate $\mathcal{M}_1$ move by move, but it has
to determine whether $w$ is accepted by $\mathcal{M}_1$ within $2^{c \times S(|w|)}$ moves. (Proof cont’d)
Proof (cont’d)

Define the procedure \( \text{reach}(\alpha, \beta, k) \) on the right: \( \alpha, \beta \) are configurations (words of input and working tapes together with the current state and the locations of tape-heads) of \( M_1 \), \( k \) is a non-negative integer, and \( C \) is initially the set of all configurations of \( M_1 \) whose length is less than or equals to \( S(|w|) + |w| \). Let \( \alpha_0 \) and \( \alpha_{\text{fin}} \) be the initial and final configurations. Given \( w \), by assumption, they are unique. If \( \text{reach}(\alpha_0, \alpha_{\text{fin}}, 2^{c \times S(|w|)}) = \text{"yes,} \) there exists a sequence \( \alpha_0, \alpha_1, \ldots, \alpha_n (= \alpha_{\text{fin}}) \) such that \( M_1 \) moves from \( \alpha_i \) to \( \alpha_{i+1} \), or \( \alpha_i = \alpha_{i+1} \), so \( w \) is accepted by \( M_1 \). If \( \text{reach}(\alpha_0, \alpha_{\text{fin}}, 2^{c \times S(|w|)}) = \text{"no,} \) \( w \) is not accepted by \( M_1 \). For \( \text{reach}(\alpha_0, \alpha_{\text{fin}}, 2^{c \times S(|w|)}) \), there are recursive calls in the program at most \( \log_2 2^{c \times S(|w|)} = c \times S(|w|) \) times. Each stack frame requires at most \( 2^{k'} \) space. Hence, this program can be realized by \( k'-\text{TM} \) using \( 2c \times S(|w|)^2 \) space. \( \square \)

Corollary

\[ \mathbb{P} \subseteq \mathbb{NP} \subseteq \mathbb{PSPACE} = \mathbb{NPSPACE} \]

Exercise

1. Show that if languages \( A, B \in \mathbb{P} \), then \( A \cup B, A \cap B, (A)^c, A \cdot B \in \mathbb{P} \).
2. Show that if languages \( A, B \in \mathbb{NP} \), then \( A \cup B, A \cap B, A \cdot B \in \mathbb{NP} \).
3. Show that for all languages \( A, B, C \) over \( \Sigma \), \( A \leq^p_m A \) (reflexivity), \( A \leq^p_m B \ & \ B \leq^p_m C \Rightarrow A \leq^p_m C \) (transitivity), \( A \leq^p_m B \Rightarrow (A)^c \leq^p_m (B)^c \).
4. Which of the following statements hold? Explain the reason why.
   (a) \( A \leq^p_m B \ & \ B \in \mathbb{P} \Rightarrow A \in \mathbb{P} \)
   (b) \( A \leq^p_m B \ & \ B \in \mathbb{NP} \Rightarrow A \in \mathbb{NP} \)
   (c) \( A \leq^p_m B \ & \ B \in \mathbb{PSPACE} \Rightarrow A \in \mathbb{PSPACE} \)
   (d) \( A \leq^p_m B \Rightarrow A \leq^p_m (B)^c \)
5. Select one of the problems on page 16 (or find another example), and then show that this problem is \( \mathbb{NP} \)-complete.
6. Show that \( \mathbb{P} = \mathbb{NP} \) if and only if a finite language is \( \mathbb{NP} \)-complete.
7. Show that the following problem is \( \mathbb{NP} \)-complete: Given a finite set \( U \), subsets \( S_1, \ldots, S_n (\subseteq U) \) and an integer \( k \), the question if there exist \( k \) subsets \( S_{i_1}, \ldots, S_{i_k} \) such that \( S_{i_1} \cup \cdots \cup S_{i_k} = U \).
Appendix: What if $P = NP$ ...

Most of researchers believe $P \neq NP$ today. But if this long-standing open question “$P = NP$?” is positively solved, what happens? : 

It turns out

- $NP = coNP$, $P = PH$ (the union of all complexity classes in the polynomial hierarchy), and thus, polynomial hierarchy is collapsed

Furthermore

- SSL, RSA, PGP are no longer secure infrastructures, so
- E-commerce business is exposed to serious menace of security flaws

On the other hand

- better predictions of weather, earthquakes and other natural phenomena are established
- mathematicians could be replaced by efficient theorem-discovering programs (Gödel 1956).

Additionally

- one who has solved the question first receives the prize of $1M from CMI [1]

VII. Parikh’s theorem
Commutative image

Let \( L \) : language over \( T \) (terminals)

\[ c(L) \text{ is the commutative image of } L \text{ if } \forall u \in \Sigma^*, \exists u \in c(L) \text{ iff } \exists w \in L : u, w \text{ are equivalent under the axiom } xy \approx yx \]

Let \( T = \{a_1, \ldots, a_n\} \)

\( \sharp_{a_i}(u) \) is the number of occurrences of \( a_i \) in a word \( u \)

\( \sharp_T(u) \) is the vector \((\sharp_{a_1}(u), \ldots, \sharp_{a_n}(u))\)

\( \Psi_T(L) \) is \( \{\sharp_T(u) \mid u \in L\} \), called Parikh image of \( L \)

Note

- \( \Psi_T(L) = \Psi_T(c(L)) \)
- \( \Psi_T(L) = \Psi_T(M) \) if and only if \( c(L) = c(M) \)

\[ \therefore \sharp_T(u) = \sharp_T(w) \text{ if and only if } u, w \text{ are equivalent under the axiom } xy \approx yx \]

Remark

\( \mathbb{N}^{[T]} \) (vectors) and \( T^*/\approx \) (commutative words) are isomorphic:

\[ \begin{align*}
\Psi_T & \downarrow \quad g(v) = \bigcup_{i=1}^{[T]} \{a_i, \ldots, a_i\}_{\text{mul}}^{v(i)} \quad g(u+v) = g(u) \cup_{\text{mul}} g(v) \\
\mathbb{N}^{[T]} & \leftrightarrow \quad \text{mul}(T) \\
\end{align*} \]

\( \text{mul}(A) : \) set of multisets over \( A \), \( \cup_{\text{mul}} : \) multiset union, \( \{\}_{\text{mul}} : \) multiset
Non-negative vector addition systems (NNVAS)

NNVAS $V = (c, \{v_1, \ldots, v_k\})$ on $\mathbb{N}^n$

- $c$ : vector in $\mathbb{N}^n$, called constant
- $v_1, \ldots, v_k$ : vectors in $\mathbb{N}^n$, called periods

* Originally, VAS [1] is equipped with vectors $v_1, \ldots, v_n$ from $\mathbb{Z}^n$ as periods.

Predicate $\Phi_V$ of NNVAS $V$

$\Phi_V(v) \iff \exists x_1, \ldots, x_k, \in \mathbb{N}: v = c + (x_1 \times v_1) + \cdots + (x_k \times v_k)$

Set $[V]$ generated by NNVAS $V$

$[V] = \{ v \in \mathbb{N}^n | \Phi_V(v) \}$

**Note**

$\Phi_V(v)$ if and only if $v \in [V]$ (decidability is decidable, so $v \notin [V]$ is decidable)


Semi-linear sets

$S$ is a linear set if $\exists$ NNVAS $V = (c, \{v_1, \ldots, v_k\})$ such that $S = [V]$

$S_1: \left( \begin{array}{c} 1 \\ 1 \end{array} \right) + x_1 \left( \begin{array}{c} 1 \\ 1 \end{array} \right)$

$S_2: \left( \begin{array}{c} 2 \\ 1 \end{array} \right) + x_1 \left( \begin{array}{c} 1 \\ 0 \end{array} \right) + x_2 \left( \begin{array}{c} 1 \\ 1 \end{array} \right)$

Semi-linear set is a finite union $S_1 \cup \cdots \cup S_n$ of linear sets

**Note**

- Linear sets are not closed under any Boolean operation. (Exercise)
- The class of semi-linear sets properly includes that of linear sets.
Semi-linear sets are closed under Boolean operations

Proof of \( \cup \)

Obvious by definition of semi-linearity.

Proof of \( \cap \)

Let \( U = (c, \{ u_1, \ldots, u_m \}) \) and \( V = (d, \{ v_1, \ldots, v_n \}) \). Define
\[
A = \{(x_1, \ldots, x_m, y_1, \ldots, y_n) \in \mathbb{N}_{m+n} \mid c + \sum_{i \leq j \leq m} x_i u_i = d + \sum_{i \leq j \leq n} y_j v_j \}
\]
\[
B = \{(x_1, \ldots, x_m, y_1, \ldots, y_n) \in \mathbb{N}_{m+n} \mid \sum_{i \leq j \leq m} x_i u_i = \sum_{i \leq j \leq n} y_j v_j \}.
\]

One can compute the sets \( S_A, S_B \) of minimal positive elements of \( A \) and \( B - 0 \), respectively, and \( S_A \) and \( S_B \) are finite (Appendix (C)-(1)), where 0 is the vector containing only 0. For each \( s \in S_A \), define an NNVAS \( W_s = (s, S_B) \). We show that \( A = \bigcup_{s \in S_A} [W_s] \) in the following.

For "\( \supseteq \)" use the induction on (the structure of) \( W_s \). The base case is obvious, because \( s \) is a minimal element of \( A \). For the induction step, suppose \( p \in S_B \) and \( q \in [W_s] \) where \( q = c + \sum_{i \leq m} q(i) u_i = d + \sum_{i \leq j \leq n} q(m+j) v_j \). Since \( p \) is a minimal element of \( B \), it satisfies \( \sum_{i \leq m} p(i) u_i = \sum_{i \leq j \leq n} p(m+j) v_j \). Then, \( p + q = c + \sum_{i \leq m} (p(i) + q(i)) u_i = d + \sum_{i \leq j \leq n} (p(m+j) + q(m+j)) v_j \). Hence, \( p + q \in A \).

For "\( \subseteq \)" suppose \( p \in A \). From minimality, \( q \leq p \) for some \( q \in S_A \). (Proof cont’d) 6

Proof of \( \cap \) (cont’d)

This implies that: \( \sum_{i \leq m} (p(i) - q(i)) u_i = \sum_{i \leq m} p(i) u_i - \sum_{i \leq m} q(i) u_i = (d-c) + \sum_{i \leq j \leq n} p(m+j) v_j - ((d-c) + \sum_{i \leq j \leq n} q(m+j) v_j) = \sum_{i \leq j \leq n} p(m+j) v_j - \sum_{i \leq j \leq n} q(m+j) v_j \), and thus, \( p - q \in B \). Observe that \( B = \{ (0, S_B) \} \): "\( \supseteq \)" is obvious, and "\( \subseteq \)" is shown by structural induction. Since \( q + (p - q) \in [W_q] \), \( p \in [W_q] \).

Let \( f \) be the function from \( \mathbb{N}_{m+n} \) to \( \mathbb{N}^m \) such that \( f(p) = \sum_{i \leq m} p(i) u_i \). Obviously, \( f(p + q) = f(p) + f(q) \), so \( f \) is a linear function (page 21). Since semi-linearity is closed under linear mapping, \( \{ f(w) \mid w \in A \} \) is semi-linear (Appendix (A)-(2)), and thus, \( [U] \cap [V] = \{ c + f(w) \mid w \in A \} \) is semi-linear.  

Proof sketch of \( (\cap) \)

Suppose that \( S \) is a finite union of linear sets \( T_1, \ldots, T_n \). By de Morgan’s law, \( (S)^c = (T_1)^c \cap \cdots \cap (T_n)^c \). So it suffices to show that the complement of a linear set is semi-linear. Let \( V = (c, \{ v_1, \ldots, v_n \}) \) be an NNVAS on \( \mathbb{N}^k \), and define \( V_0 = (0, \{ v_1, \ldots, v_n \}) \).

Then, (1) \( (V_0)^c \) is semi-linear (Appendix (B)), (2) \( X = \{ x \in \mathbb{N}^k \mid c \not\leq x \} \) is semi-linear, (3) let \( Y = \{ y \in \mathbb{N}^k \mid c \leq y \} \), then \( (Y)^c = X \cup (Y - [V]) \). Let \( f \) be the function of \( \mathbb{N}^k \) such that \( f(x) = x + c \). Since \( f \) is a bijective function of \( \mathbb{N}^k \) onto \( Y \), \( Y - [V] \) is semi-linear if and only if \( f^{-1}(Y - [V]) \) is semi-linear. Observe that \( f^{-1}(Y - [V]) = f^{-1}(Y) - f^{-1}([V]) = N^k - [V_0] \). According to (1) & (2), \( f^{-1}(Y - [V]) \), which is \( (V_0)^c \), is semi-linear, and hence, \( X \cup (Y - [V]) \) is semi-linear.  

Parikh’s theorem

Given CFG $\mathcal{G} = (\Sigma, T, N, q_0, \Delta)$,

1. there exist NNVAS’s $V_1, \ldots, V_k$ such that $\Psi_T(\mathcal{L}(\mathcal{G})) = \bigcup_{1 \leq i \leq k} \{ V_i \}$, and $V_i$ ($1 \leq i \leq k$) is effectively computable from $\mathcal{G}$
   (Parikh image of context-free language is effectively semi-linear)

2. there exists a regular language $L$ such that $c(\mathcal{L}(\mathcal{G})) = c(L)$.
   (Commutative images of the classes of CFL and RL coincide)

Proof

First we show (1): For each $Q \subseteq N$, define $\mathcal{L}_Q(\mathcal{G}) = \{ w \in \Sigma^* \mid w$ is obtained from a derivation tree in which every non-terminal symbol of $N$ appears $\}$. Observe that $\mathcal{L}(\mathcal{G}) = \bigcup_{Q \subseteq N} \mathcal{L}_Q(\mathcal{G})$, and thus, $\Psi_T(\mathcal{L}(\mathcal{G})) = \bigcup_{Q \subseteq N} \Psi_T(\mathcal{L}_Q(\mathcal{G}))$. Define the conditions
   (a) all $q$ in $Q$ occur in the tree,
   (b) no $q$ in $Q$ occurs more than $|N|$-times on any path from the root to a leaf.

And then, define the sets $D_Q, \bar{D}_Q$ of derivation trees whose root is $q_0$:

$D_Q = \{ t \mid$ derivation tree $t$ whose leaves are terminals and that satisfies (a) & (b) $\}$
$\bar{D}_Q = \{ t \mid$ derivation tree $t$ whose leaves are terminals and that satisfies (a) $\}$

(Proof cont’d)

Moreover, define the set $I_Q$ as follows:

$I_Q = \{ t \mid$ derivation tree $t$ whose root is $q \in Q$ and whose leaves contain exactly
one non-terminal $q$ and that satisfies (b) $\}$

Let $V_Q = \bigcup_{s \in D_Q} [\langle \sharp_T(\text{leaf}(s)), \{ \sharp_T(\text{leaf}(t)) \mid t \in I_Q \} \rangle]$, then we show that for each $Q \subseteq N$, $V_Q = \Psi_T(\mathcal{L}_Q(\mathcal{G}))$. For “$\subseteq$”, use the induction on vectors in $V_Q$. In the base case, consider some $s \in D_Q$ such that $\sharp_T(\text{leaf}(s))$ is a constant in $V_Q$. By definition, $\text{leaf}(s) \in \mathcal{L}_Q(\mathcal{G})$, and thus, $\sharp_T(\text{leaf}(s)) \in \Psi_T(\mathcal{L}_Q(\mathcal{G}))$.

For induction step, suppose $v_1 \in V_Q$ and $v_2 = \sharp_T(\text{leaf}(t))$ for some $t \in I_Q$. By induction hypothesis, $v_1 \in \Psi_T(\mathcal{L}_Q(\mathcal{G}))$, and thus, there exists a derivation tree $u \in \bar{D}_Q$ such that $\text{leaf}(u) \in \mathcal{L}_Q(\mathcal{G})$ and $\sharp_T(\text{leaf}(u)) = v_1$. If the root of $t$ is labeled by $q \in Q$ (so $\text{leaf}(t) = w_1 q w_2$), then $t = C[q]$. Since $u = C'[u']$ for some $u'$ such that the root of $u'$ is labeled by $q$, the tree $C'[C[u']]$ (obtained by inserting $C$ in between $C'$ and $u'$) is a derivation tree in $\bar{D}_Q$. Since $\sharp_T(\text{leaf}(C'[C[u']]))) = \sharp_T(\text{leaf}(C'[u'])) + \sharp_T(\text{leaf}(C)) = v_1 + \sharp_T(\text{leaf}(t)) = v_1 + v_2$, we obtain $v_1 + v_2 \in \Psi_T(\mathcal{L}_Q(\mathcal{G}))$.

Next, for “$\supseteq$”, use the induction on trees in $\bar{D}_Q$. In the base case, consider $t \in D_Q$. By definition, $\sharp_T(\text{leaf}(t)) \in V_Q$. For induction step, suppose $t \in \bar{D}_Q$ and $t \notin D_Q$ such that every tree $u$ in $\bar{D}_Q$ smaller than $t$ satisfies $\sharp_T(u) \in V_Q$.

(Proof cont’d)
By assumption, \( t \) has a path (from the root to a leaf) that contains a non-terminal \( q \in Q \) occurring more than \(|Q|\)-times. Let \( n = |Q| \). Then, \( t = C[C_1[\cdots C_m[u]]\cdots] \) such that the root of \( C_i \) \((1 \leq i \leq m \text{ and } n < m)\) and the root of \( u \) are \( q \). See the right figure. Here \( C \) is possibly the empty context. We will obtain a smaller tree from \( t \) by removing a context among \( C_1, \ldots, C_m \). If some of non-terminals in \( Q \) appears only in \( C_i, C_i \) cannot be a candidate, because the resulting tree is not in \( D_Q \). However, since \(|Q - \{q\}| = n - 1 < m \), there is at least a context, say \( C_{\ell} \) (yellow part), such that \( t' \in D_Q \) where \( t' = C[C_1[\cdots C_{\ell-1}[C_{\ell+1}[\cdots C_m[u]]\cdots]]] \). By induction hypothesis, leaf(\( t' \)) \( \in V_Q \). If \( C_i \) satisfies the condition (b), leaf(\( t' \)) + leaf(\( C_i \)) \( \in V_Q \), because leaf(\( C_i \)) \( \in I_Q \). If there is no such context in \( C_1, \ldots, C_m \), find another non-terminal from \( Q \) that occurs more than \(|Q|\)-times in the same root-leaf path, because this path does not satisfy the condition (b). Repeating this process, one can eventually find a context satisfying (b).

For (2), take the regular grammar \( \mathcal{G}_V \) with production rules \( q_0 \rightarrow a_1^{(1)} \cdots a_n^{(n)} \mid a_1^{v_1(1)} \cdots a_n^{v_n(n)} q_0 \mid \cdots \mid a_1^{v_1(n)} \cdots a_n^{v_n(n)} q_0 \) for \( \text{NNVAS } V = (c, \{v_1, \ldots, v_k\}) \) on \( \mathbb{N}^n \), where \( v_i(j) \) is \( j \)-th element of vector \( v_i \), then \( \Psi_T(\mathcal{L}(\mathcal{G}_V)) = [V] \). This implies that for every semi-linear set \( S \), there exists a regular grammar \( \mathcal{G} \) such that \( \Psi_T(\mathcal{L}(\mathcal{G})) = S \). \( \square \)

### Language ineqations

Let \( \Sigma \) : alphabet with \( T = \{a_1, \ldots, a_m\} \) and \( N = \{x_1, \ldots, x_n\} \),

\[ L_\Sigma \] is a set of language components over \( \Sigma \):

\[ \varepsilon, a_1, \ldots, a_m, x_1, \ldots, x_n, \bot, uw, u + w \in L_\Sigma \text{ if } u, w \in L_\Sigma \]

\[ f(x_1, \ldots, x_n) \leq x_i \text{ is a language inequation if } f(x_1, \ldots, x_n) \in L_\Sigma \]

Let \( L_1, \ldots, L_n \) : languages over \( T \),

\( f(x_1, \ldots, x_n) \) : language components over \( \Sigma \)

\[ [f(x_1, \ldots, x_n)](L_1, \ldots, L_n) \text{ is value of } f(x_1, \ldots, x_n) : \]

\[ [\varepsilon](L_1, \ldots, L_n) = \{\varepsilon\} \quad [\bot](L_1, \ldots, L_n) = \emptyset \]

\[ [a_i](L_1, \ldots, L_n) = \{a_i\} \quad [x_i](L_1, \ldots, L_n) = L_i \]

\[ [uw](L_1, \ldots, L_n) = [u](L_1, \ldots, L_n) \cdot [w](L_1, \ldots, L_n) \]

\[ [u + w](L_1, \ldots, L_n) = [u](L_1, \ldots, L_n) \cup [w](L_1, \ldots, L_n) \]
Solutions of language inequations

Let \( f_i(x_1, \ldots, x_n) \leq x_i \) : language inequations over \( \Sigma \) (\( 1 \leq i \leq n \))

\( L_1, \ldots, L_n \) : languages over \( T \)

\( (L_1, \ldots, L_n) \) is a solution of \( f_i(x_1, \ldots, x_n) \leq x_i \)

if \( L_i \) is a minimal language satisfying \([f_i(x_1, \ldots, x_n)](L_1, \ldots, L_n) \subseteq L_i\)

Note 1 (Ginsburg & Rice)

\( (L_1, \ldots, L_n) \) is a solution of \( f_i(x_1, \ldots, x_n) \leq x_i \) (\( 1 \leq i \leq n \)) iff \( L_i = L(G_i) \) such that \( G_i = (\Sigma \cup \{ \perp \}, P, \{ x_1, \ldots, x_n, \perp \}, x_i, \{ x_i \to f_i(x_1, \ldots, x_n) | 1 \leq i \leq n: f_i(x_1, \ldots, x_n) \leq x_i \})\)

Note 2

We say \( (L_1, \ldots, L_n) \) is a solution of \( f_i(x_1, \ldots, x_n) \leq x_i \) over a commutative alphabet if \( L_i \) is a minimal language satisfying \( c([f_i(x_1, \ldots, x_n)](L_1, \ldots, L_n)) \subseteq c(L_i) \). In this definition, \( L_i \) in \( (L_1, \ldots, L_n) \) is always a regular language \( (1 \leq i \leq n) \). However, this is not a consequence of Parikh’s theorem. (Proof is explained later)


Commutative Kleene algebra

Commutative Kleene algebra with variables \( X \) is \( (A, X, \{+, \cdot, *, 1, 0\}) \)

\( A \) : carrier set

\( X \) : finite set of variables

such that the following axioms hold for operators:

[Associativity] \( (x + y) + z = x + (y + z) \)

[Commutativity of +] \( x + y = y + x \)

[Commutativity of \( \cdot \)] \( x \cdot y = y \cdot x \)

[Distributivity] \( x \cdot (y + z) = x \cdot y + x \cdot z \)

[Identity of +] \( x + 0 = x \)

[Identity of \( \cdot \)] \( x \cdot 1 = x \)

[Idempotency] \( x + x = x \)

[Nullpotency] \( x \cdot 0 = 0 \)

[Kleene star] \( 1 + x \cdot x^* = x^* \)

Corollary

\( (x + y)^* = x^* \cdot y^* \quad x + y \cdot z \leq z \Rightarrow x \cdot y^* \leq z \quad (x \leq y \iff x + y = y) \)
Let $K[X]$: commutative Kleene algebra with variables $X$

$D$ is mapping from $K[X]$ to $K[X]$ such that (called differential operator)

$D(x + y) = D(x) + D(y)$
$D(x \cdot y) = x \cdot D(y) + y \cdot D(x)$
$D(x^* ) = x^* \cdot D(x)$
$D(1) = D(0) = 0$

$\frac{\partial}{\partial x}$ is differential operator for $x \in X$ such that

$\frac{\partial x}{\partial x} = 1$
$\frac{\partial y}{\partial x} = 0$ if $y \in X - \{x\}$
$\frac{\partial a}{\partial x} = 0$ if $a \in A$

$\frac{\partial}{\partial x}(f(e)) = \frac{\partial f}{\partial x}(e) \cdot \frac{\partial e}{\partial x}$ ($\frac{\partial f}{\partial x}$ is denoted by $f'(x)$)

Solution of $f_i(x_1, \ldots, x_n) \leq x_i$ in $K[X]$

Let $f(x_1, \ldots, x_n)$: finite expression in $K[X]$
$e_1, \ldots, e_n$: finite expressions in $K[\emptyset]$

$(e_1, \ldots, e_n)$ is solution of $f(x_1, \ldots, x_n) \leq x_i$ in $K[X]$
if $e_i$ is a minimal subset of $A$ satisfying $f(e_1, \ldots, e_n) \subseteq e_i$

Proposition (Hopkins & Kozen 1999)

Every $f(x) \leq x$ in $K[\{x\}]$ has the unique solution $f'(f(0))^* \cdot f(0)$

Proof

First, we observe that for all polynomials $e, g, h, k$ in $K[X],$

1. $e(x + g) = e(x) + e'(x + g) \cdot g$
   $e(g) = e(0) + e'(g) \cdot g$ if $x = 0$

2. $e \cdot h \leq g \cdot h \Rightarrow k(e) \cdot h \leq k(g) \cdot h$

Each statement can be shown by the induction on the structure of polynomials.

(Proof cont’d)
Proof (cont’d)

Let \( e = g \cdot h, \ g = f(0), \ h = f'(g)^* \) in the previous (2), then \( e \cdot h = f(0) \cdot f'(g)^* \cdot f'(g)^* = f(0) \cdot f'(g)^* \) and \( g \cdot h = f(0) \cdot f'(g)^* \). Thus, for any polynomial \( k, \ k(g \cdot h) \cdot h \leq k(g) \cdot h \) in this case. Therefore, one can conclude that \( f'(f(0)) \cdot f(0) \) satisfies \( f(x) \leq x : \)

\[
f(f'(f(0))^* \cdot f(0)) = f(g \cdot h)
\]

\[
= f(0) + f'(g \cdot h) \cdot g \cdot h \quad \text{by (1)}
\]

\[
\leq f(0) + f'(g) \cdot g \cdot h \quad \text{by the above observation}
\]

\[
= g + f'(g) \cdot g \cdot f'(g)^*
\]

\[
= (1 + f'(g) \cdot f'(g)^*) \cdot g
\]

\[
= f'(g)^* \cdot g \quad \text{by [Kleene star]}
\]

\[
= f'(f(0))^* \cdot f(0)
\]

For the "least" solution, we show that for every polynomial \( k \) in \( K[[x]] \) that satisfies \( f(k) \leq k, \ f'(f(0))^* \cdot f(0) \leq k \). According to Corollary in page 13, it suffices to show that \( f(0) + f'(f(0)) \cdot k \leq k \) : From monotonicity \( f(0) \leq f(k) \) (as \( 0 \leq k \)) and the assumption \( f(k) \leq k \), one can have \( f'(f(0)) \leq f'(k) \). Thus,

\[
f(0) + f'(f(0)) \cdot k \leq f(0) + f'(k) \cdot k
\]

\[
\leq f(k) \quad \text{by (1)}
\]

\[
\leq k \quad \text{by assumption}
\]

Hence, uniqueness is justified by the fact that \( f'(f(0))^* \cdot f(0) \) is the least solution. \( \Box \)

---

Corollary (Generalization of Parikh’s theorem)

Every system of inequations \( f_i(x_1, \ldots, x_n) \leq x_i \ (1 \leq i \leq n) \) in \( K[X] \) has the unique solution, and is effectively computable from \( f_1, \ldots, f_n \)

---

Proof

Suppose \( X = \{x, y\} \), and let the system of inequations: \( f(x, y) \leq x \) and \( g(x, y) \leq y \). First, freeze \( x \), meaning that we consider \( K[[x]][[y]] \) instead of \( K[[x, y]] \). According to the previous proposition, one can compute the (least) solution \( h(x) \) of the inequation \( g(x, y) \leq y \). And then, compute the solution of \( f(x, h(x)) \leq x \) in \( K[[x]] \). Let \( (k, h(k)) \) be the solution obtained by this computation. For the claim that \( (k, h(k)) \) is the least solution, let \( (p, q) \) be a solution of the above system. Since the least solution of \( g(p, y) \leq y \) is \( h(p) \) where \( x \) in \( g(x, y) \) is instantiated by \( p, h(p) \leq q \). By monotonicity and the assumption that \( (p, q) \) is a solution, \( f(p, h(p)) \leq f(p, q) \leq p \). Because \( k \) is the least solution of \( f(x, h(x)) \), one can conclude that \( k \leq p \). Therefore,

\[
(k, h(k)) \leq (k, h(p)) \quad \text{by monotonicity} \ (k \leq p)
\]

\[
\leq (p, q) \quad \text{by the above observation} \ (h(p) \leq q)
\]

Iteratively applying the above computation to the system \( S \) of inequations in \( K[X] \), it results in the least solution of \( S \). \( \Box \)
Example

Consider the CFG $\mathcal{G}_1, \mathcal{G}_2$ with the following production rules:

\[
\begin{align*}
\Delta_1 & : 
q_0 \rightarrow aq_0b & q_0 \rightarrow \varepsilon \\
\Delta_2 & : 
q_0 \rightarrow aq_1 & q_0 \rightarrow bq_2 & q_0 \rightarrow \varepsilon \\
& q_1 \rightarrow aq_1 & q_1 \rightarrow bq_0 \\
& q_2 \rightarrow aq_0 & q_2 \rightarrow bq_2q_2
\end{align*}
\]

Then $\mathcal{L}(\mathcal{G}_1)$ and $\mathcal{L}(\mathcal{G}_2)$ are solutions for $x$ of the following systems, respectively:

\[
\begin{align*}
S_1 & : 
abx + 1 \leq x \\
S_2 & : 
ay + bz + 1 \leq x \quad ay^2 + bx \leq y \quad ax + bz^2 \leq z
\end{align*}
\]

For $S_1$, according to \textit{Proposition (Hopkins & Kozen 1999)}], let $f(x) = abx + 1$, $f'(f(0))^* \cdot f(0) = (ab)^* \cdot 1 = (ab)^*$ that is equivalent to $\mathcal{L}(\mathcal{G}_2)$ under commutativity.

For $S_2$, let $f(x, y, z) = ay + bz + 1$, $g(x, y, z) = ay^2 + bx$, $h(x, y, z) = ax + bz^2$. First, freeze $x, y$, meaning that consider $K\{x\}\{y\}\{z\}$ for $K\{x, y, z\}$: Let $\ell_h(z) = ax + bz^2$, then $z = \ell_h(\ell_h(0))^* \cdot \ell_h(0) = (abx)^* \cdot ax$. Next, let $\ell_g(y, z) = ay^2 + bx$, and then consider $\ell_g(y, (abx)^* \cdot ax) = ay^2 + bx$. Similar to the previous step, we obtain $y = (abx)^* \cdot bx$. Finally, consider $f(x, (abx)^* \cdot bx, (abx)^* \cdot ax) = (abx) \cdot (abx)^* + 1 = (abx)^*$. Let $p(x) = (abx)^*$, then computing $p'(x)$ and $p'(p(0))^* \cdot p(0)$ is \textit{Exercise}.

Exercise

1. Show that the commutative image of a context-free language is not context-free.
2. Construct examples showing the claim in page 5 that the class of linear sets is not closed under union, intersection or complement.
3. Show that semi-linearity is closed under projection, meaning that for every projection $f_i$ such that $f_i(v) = (v(1), \ldots, v(i - 1), v(i + 1), \ldots, v(k))$, if $S$ is a semi-linear subset of $\mathbb{N}^k$, then $f_i(S) = \{f_i(v) \mid v \in S\}$ is semi-linear.
4. Show that semi-linearity is closed under $\times$, meaning that if $S$ and $T$ are semi-linear subsets of $\mathbb{N}^m$ and of $\mathbb{N}^n$, then $S \times T$ is semi-linear.
5. Construct an example showing that for a language $L$ over $T$, $\Psi_T(L)$ is semi-linear but $L$ is not context-free.
6. Show that $\{ w \in \{a, b\}^* \mid |w|_a = (|w|_b)^2 \}$ is not context-free.
7. Show that every context-free language over a one-letter alphabet is regular.
8. Compute $p'(x)$ and the solution for $x, y, z$ of $S_2$ in page 18.
Appendix (A) : Basic properties of semi-linear sets

(1) Every linear set on $\mathbb{N}^k$ is a finite union of linear sets on $\mathbb{N}^k$, each of which is linearly independent periods.

Proof
Use the induction on the number of periods. The base case is obvious, because a linear set (obtained by an NNVAS) with one period satisfies (1). For induction step, let $L = \{V \mid V = (c, \{v_1, \ldots, v_n\})\}$, and suppose $v_1, \ldots, v_n$ is linearly dependent. Then, there exist a permutation $\pi$ over $\{1, \ldots, n\}$, non-negative integers $t_i (1 \leq i \leq k)$ and positive integers $t_j (k < j \leq n)$ such that $\sum_{1 \leq i \leq k} t_i v_{\pi(i)} = \sum_{k+1 \leq j \leq n} t_j v_{\pi(j)}$ (*). For each $j$ ($k < j \leq n$), define $C_j = \{c + xv_{\pi(j)} \mid 0 \leq x < t_j\}$ and $P_j = \{v_1, \ldots, v_n\} - \{v_{\pi(j)}\}$. Let $L_j = \bigcup_{0 \leq x < t_j} \{(c + xv_{\pi(j)}, P_j)\}$, then by induction hypothesis, $L_j$ is the finite union of linear sets, each of which satisfies (1).

Next, we show that $L = \bigcup_{k<j \leq n} L_j$. By construction, $L_j \subseteq L (k < j \leq n)$, and thus, $\bigcup_{k<j \leq n} L_j \subseteq L$. For “$\subseteq$”, let $v = c + \sum_{1 \leq i \leq n} d_i v_{\pi(i)}$ in $\mathbb{N}^k$. If $d_j \geq t_j (k < j \leq n)$, then take $u_1 = c + \sum_{1 \leq i \leq k} (d_i + t_i) v_{\pi(i)} + \sum_{k<j \leq n} (d_j - t_j) v_{\pi(j)}$. From (*), $v = u_1$. After $\ell$-times application of the above procedure, one can obtain $u_\ell$ such that $v = u_\ell$ and a coefficient of $v_{\pi(j)}$ is less than $t_j$ for some $j (k < j \leq n)$. Let $u_\ell = c + \sum_{1 \leq i \leq n} e_i v_{\pi(i)}$, then $u_\ell \in L_j$, because $c + e_j v_{\pi(j)} \in C_j$ and $\{v_1, \ldots, v_n\} - \{v_{\pi(j)}\} = P_j$. □

As a corollary of (1), it follows that: every semi-linear set on $\mathbb{N}^k$ is a finite union of linear sets on $\mathbb{N}^k$, each of which is linearly independent periods.

Appendix (A) : Basic properties of semi-linear sets (cont’d)

A function $f$ from $\mathbb{N}^m$ to $\mathbb{N}^n$ is linear if $f(x + y) = f(x) + f(y)$ :

(2) Semi-linearity is closed under linear mapping, meaning that for every linear function $f$ from $\mathbb{N}^m$ to $\mathbb{N}^n$, if $S$ is a semi-linear subset of $\mathbb{N}^m$, then $f(S) = \{f(v) \mid v \in S\}$ is semi-linear.

Proof
It suffices to show that linearity is closed under linear mapping. Let $L = \{V \mid V \subseteq \mathbb{N}^m\}$ where $V = (c, \{v_1, \ldots, v_k\})$, then $f(L) = \{V' \mid V' = (f(c), \{f(v_1), \ldots, f(v_k)\})\}$. So, for every $x \in L$, $f(x) \in f(L)$. Conversely, if $u \in f(L)$, then $u = f(c) + \sum_{1 \leq i \leq k} y_i f(v_i) = f(c + \sum_{1 \leq i \leq k} y_i v_i)$. Hence, $u = f(z)$ for some $z \in L$. □

(3) Semi-linearity is closed under inverse linear-mapping, meaning that if $S$ is a semi-linear subset of $\mathbb{N}^m$, then $f^{-1}(S) = \{v \in \mathbb{N}^m \mid \exists f(v) \in S\}$ is semi-linear.

Proof
For $p = (x_1, \ldots, x_a)$ and $q = (y_1, \ldots, y_b)$, we denote $p \times q$ for $(x_1, \ldots, x_a, y_1, \ldots, y_b)$. Let $g$ be the function $g(x) = x \times f(x)$. From linearity of $f$, we have $g(x + y) = (x + y) \times (f(x) + f(y)) = (x \times f(x)) + (y \times f(y)) = g(x) + g(y)$. So, $g$ is a linear function. From (2), $g(\mathbb{N}^m)$ is a semi-linear subset of $\mathbb{N}^m$. Moreover, since $\mathbb{N}^m \times S$ is semi-linear, $g(\mathbb{N}^m) \cap (\mathbb{N}^m \times S)$ is semi-linear, because semi-linearity is closed under intersection. Let $h$ be the projection $h(x \times y) = x$, then $h(g(\mathbb{N}^m) \cap (\mathbb{N}^m \times S)) = f^{-1}(S)$. Hence, $f^{-1}(S)$ is semi-linear, because semi-linearity is closed under projection. □
Appendix (B) : Complement of semi-linear sets

For every NNVAS $V = (0, \{v_1, \ldots, v_n\})$ on $\mathbb{N}^k$ with linearly independent vectors $v_1, \ldots, v_n$, $([V])^c$ is semi-linear. ($0$ is the vector containing only 0)

Proof

If $k > n$, then find a mapping $\pi$ from $\{1, \ldots, k-n\}$ to $\{1, \ldots, k\}$ such that for unit vectors $e_{\pi(1)}, \ldots, e_{\pi(k-n)}$ (each $e_i$ of which contains exactly one 1 at $i$-th position and the other elements are 0), $v_1, \ldots, v_n, e_{\pi(1)}, \ldots, e_{\pi(k-n)}$ are linearly independent. So one can take $V' = (0, \{v_1, \ldots, v_k\})$ and $v_1, \ldots, v_k$ are linearly independent. Then, it holds that: there exists a positive integer $\ell_V$ for $V'$ such that $\mathbb{N}^k = \{u \mid \exists x \in \mathbb{N}, \exists y_1, \ldots, y_k \in \mathbb{Z} : 1 \leq x \leq \ell_V \& x u = \sum_{1 \leq i \leq k} y_i v_i\}$. First, we show that for each subset $I$ of $\{1, \ldots, k\}$, $S_I = \{u \times (y_1, \ldots, y_k) \mid \exists x, y_1, \ldots, y_k \in \mathbb{N} : xu + \sum_i q_i v_i = \sum_{j \notin I} y_j v_j\}$ is effectively semi-linear. Note that $S_I \subseteq \mathbb{N}^{2k}$. Define the function $f_{I,x}$ of $\mathbb{N}^{2k}$ such that $f_{I,x}(p \times q) = (xp + \sum_{i \in I} q_i (i) v_i) \times \sum_{j \notin I} q(j) v_j$, and define $g$ from $\mathbb{N}^k$ to $\mathbb{N}^{2k}$ such that $g(r) = r \times r$. Let $F = f_{I,x}((p_1 \times q_1) + (p_2 \times q_2))$, then

$$F = \left( (xp_1 + \sum_{i \in I} q_1(i) v_i) \times \sum_{j \notin I} q_2(j) v_j \right) \times \sum_{j \notin I} q_1(i) v_i + \left( (xp_2 + \sum_{i \in I} q_2(i) v_i) \times \sum_{j \notin I} q_1(j) v_j \right) \times \sum_{j \notin I} q_1(j) v_j$$

so $f_{I,x}$ is a linear function. Moreover, $g$ is a linear function, because $g(x + y) = (x + y) \times (x + y) = (x \times x) + (y \times y) = g(x) + g(y)$. (Proof cont’d)

Appendix (B) : Complement of semi-linear sets (cont’d)

From (1), $D = \{g(p) \mid p \in \mathbb{N}^k\}$ is semi-linear. From (3), $f^{-1}_{I,x}(D)$ is semi-linear. Observe that $\bigcup_{1 \leq x \leq \ell_V} f^{-1}_{I,x}(D) = \bigcup_{u \in \mathbb{N}^{2k}} \{p \in \mathbb{N}^k \mid f_{I,x}(p) \in D\} = S_I$.

Next, for each non-empty subset $I$ of $\{1, \ldots, k\}$, define $c_I = (0, \ldots, 0, a_1, \ldots, a_k)$ where $a_i = 1$ if $i \in I$; otherwise, $a_i = 0$. Let $E_I = \{c_I, (e_1, \ldots, e_{2k})\}$ such that $e_i (1 \leq i \leq 2k)$ is the unit vector whose $i$-th element is 1, and let $h(x \times y) = x$. For each $I \subseteq \{1 \leq x \leq \ell_V\}$ with $I \neq \emptyset$, define $K_I = \bigcup_{u \in \mathbb{N}^{2k}} [E_I] \cap f^{-1}_{I,x}(D)$, then $K_I$ is semi-linear, and thus, $h(K_I)$ is semi-linear, because $h$ is a linear function. We take $T_I = h(K_I)$. Since $T_I = \{u \in \mathbb{N}^k \mid \exists x, y_1, \ldots, y_k \in \mathbb{N}: xu = \sum_{i \in I} (y_i) v_i + \sum_{j \notin I} y_j v_j \& y_i > 0 (i \in I)\}$, $T_I \cap [V]^c = \emptyset$ for all non-empty subset $I$.

Next, define an NNVAS $P_i (n < i \leq k)$ where $P_i = (e_{n+i}, (e_1, \ldots, e_{2k}))$ and $n$ is the number of periods of $V$. Since $[P]$ is semi-linear, $[P] \cap K_I$ is semi-linear for each $I \subseteq \{1, \ldots, k\} - \{i\}$. Let $U_I = \bigcup_{1 \leq x \leq \ell_V} h([P]] \cap K_I)$. If $I \subseteq \{1, \ldots, k\} - \{i\}$, then $U_I = \{u \in \mathbb{N}^k \mid \exists x, y_1, \ldots, y_k \in \mathbb{N}, \exists y \in \mathbb{Z}: xu = \sum_{i \in I} y_i v_i \& y_j > 0 \& j > n\}$. Hence, $U_I \cap [V]^c = \emptyset$.

Next, for each $x (1 \leq x \leq \ell_V)$ and $j (1 \leq j \leq n)$, where $n$ is the number of periods of $V$, let

$$Q_{x,j} = \{u \times y \mid \exists y \in \mathbb{N}^n : xu = \sum_{i \in I} y(i) v_i \& y(j) \mod x \neq 0\}$$

$$R_x = \{u \times y \mid \exists y \in \mathbb{N}^n : xu = \sum_{i \in I} y(i) v_i \& y \neq 0\}.$$

We show that for every $x$ and $j$, $Q_{x,j}$ is effectively semi-linear. The set $\min_{x,j}(R_x)$ of minimal solutions of $R_x$ is finite and computable (Appendix (C)). (Proof cont’d)
Appendix (B) : Complement of semi-linear sets (cont’d)

Observe that $R_x = \{(0, \min_\geq(R_x))\}$. Moreover,

$$Q_{x,j} = R_x \cap \{p \times y \mid \exists p \in \mathbb{N}^k, y \in \mathbb{N}^n : 1 \leq y(j) < x\}$$

Since $\{p \times y \mid \exists p \in \mathbb{N}^k, y \in \mathbb{N}^n : 1 \leq y(j) < x\}$ is effectively semi-linear, $Q_{x,j}$ is so. Let $h'$ be the function from $\mathbb{N}^{k+n}$ to $\mathbb{N}^k$ such that $h'(p \times y) = p$, then $h'(Q_{x,j})$ is semi-linear, because $h'$ is a linear function.

Finally, we show that

$$([V'])^C = \bigcup_{\varnothing \neq I \subseteq \{1, \ldots, k\}} T_I \cup \bigcup_{1 \leq i \leq k, 1 \leq j \leq n} U_I \cup \bigcup_{1 \leq x \leq t, 1 \leq j \leq n} h'(Q_{x,j}).$$

By construction, “$$” is obvious : We already verified for the first two cases. For the last case, suppose $v \in Q_{x,j}$ for some $x (1 \leq x \leq t_V)$ and $j (1 \leq j \leq n)$, which means that $x v = \sum_{1 \leq i \leq n} y(i) v_i$ and $y(j)$ cannot be divided by $x$. This implies that if $x = 1$, $y(j)$ cannot be an integer. Hence, $v \notin [V']$.

For “$$”, suppose $v \in ([V'])^C$, then there exists $x \in \mathbb{N}$, $y_1, \ldots, y_k \in \mathbb{Z}$ such that $1 \leq x \leq t_V$ and $x v = \sum_{1 \leq i \leq n} y(i) v_i$. If $y_i < 0 (1 \leq i \leq k)$, then $v \in T_I$ for some non-empty subset $I$ of $\{1, \ldots, t_V\}$. If $y_i > 0 (n + 1 \leq i \leq k)$, then $v \in U_I$ for some $1 \leq j \leq k, I \subseteq \{1, \ldots, k\} - \{j\}$. So, assume $y_i \geq 0 (1 \leq i \leq n)$. If for all $i (1 \leq i \leq n)$, $y_i$ is divided by $x$, $v \in [V']$. Thus, there exists $j$ such that $y_j$ is not divided by $x$. Hence, $v \notin h'(Q_{x,j})$.

\[ \square \]

Alternative proof is obtained by bijective correspondence to Presburger arithmetic, where negation can be eliminated, and so negation-free NNVAS formula is obtained. 24

Appendix (C) : Minimal solutions

Every semi-linear set $\bigcup_{1 \leq i \leq n} [\{c_i, \{v_{P(i)}\}\}]$ contains only finitely minimal elements $c_i (1 \leq i \leq n)$. This observation can be generalized as follows :

(1) Every set of incomparable vectors in $\mathbb{N}^k$ is finite.

**Proof**

Use the induction on $k$. The base case is obvious, because $k = 1$. For induction step, define the projection $f_k$ from $\mathbb{N}^k$ to $\mathbb{N}^{k-1}$ such that $f_k(v) = (v(1), \ldots, v(k-1))$. Suppose for leading to the contradiction that there exists an infinite subset $S$ of $\mathbb{N}^k$ whose elements are pairwise incomparable. For each $u, v \in S$, one of the following holds : (a) $f_k(u)$ and $f_k(v)$ are incomparable, (b) $f_k(u) > f_k(v)$ and $u(k) < v(k)$, (c) $f_k(u) < f_k(v)$ and $u(k) > v(k)$. By induction hypothesis, (a) holds for only finitely many pairs. If (b) holds for infinitely many pairs, there exists an infinite sequence $u_1, u_2, \ldots$ such that $f_k(u_i) < f_k(u_{i+1})$ and $u_i(k) > u_{i+1}(k)$. However, it contradicts to the well-foundedness of $>$ on $\mathbb{N}$. For the same reason, (c) does not hold for infinitely many pairs, and hence, our assumption leads to the contradiction. \[ \square \]

As a corollary of (1), it holds that : **Every** subset of $\mathbb{N}^k$ contains only finitely many minimal elements.

In contrast, it holds that : If $k \geq 2$, for every subset of $\mathbb{N}^k$ containing $m$ minimal elements, there exists a subset of $\mathbb{N}^k$ which contains more than $m$ minimal elements (the number of incomparable minimal elements in $\mathbb{N}^k (k \geq 2)$ is unbounded).
Appendix (C) : Minimal solutions (cont’d)

(2) One can compute the set \( S \) of minimal positive solutions of the equation :
\[
 w = \sum_{1 \leq i \leq m} x_i u_i - \sum_{1 \leq j \leq n} y_j v_j \quad (u_1, \ldots, u_m, v_1, \ldots, v_n \in \mathbb{N}^k, \ w \in \mathbb{Z}^k) \quad (+1)
\]

Proof

Let \( V = \{ u_i, v_j \mid 1 \leq i \leq m, \ 1 \leq j \leq n \} \). First, we show that the question if there exists a positive solution of \( V \) is decidable. If \( V \cup \{ w \} \) is linear independent, there is no solution. If \( V \) is linear independent and \( V \cup \{ w \} \) is linear dependent, then one can compute the a unique solution \( p \) over \( \mathbb{Q}^m \) and \( q \) over \( \mathbb{Q}^n \) such that \( w = \sum_{1 \leq i \leq m} p(i) u_i - \sum_{1 \leq j \leq n} q(j) v_j \), which means that the equation has the positive solution over \( \mathbb{N}^{m+n} \) if and only if \( p \in \mathbb{N}^m \) and \( q \in \mathbb{N}^n \). Suppose that \( V \) is linear dependent. The following proof proceeds by induction on \( m + n \). The base case (the case of \( m + n = 1 \)) is obvious, because there is no such \( V \) (the above equation forms \( w = x_1 u_1 \) or \( w = -y_1 v_1 \)). For induction hypothesis, observe that one can compute subsets \( I \subseteq \{ 1, \ldots, m \} \) and \( J \subseteq \{ 1, \ldots, n \} \) and vectors \( p \in \mathbb{N}^m \) and \( q \in \mathbb{N}^n \) such that
\[
 \sum_{i \in I} p(i) u_i - \sum_{j \in J} q(j) v_j = \sum_{i \not\in I} p(i) u_i - \sum_{j \not\in J} q(j) v_j
\]
with either \( p(i) > 0 \) for some \( i \in I \) or \( q(j) > 0 \) for some \( j \in J \). This implies that \((+1)\) has a positive solution \( x_i, y_j \ (1 \leq i \leq m, \ 1 \leq j \leq n) \) if and only if it satisfies either \( x_i \leq p(i) \) for some \( i \in I \) or \( y_j \leq q(j) \) for some \( j \in J \). (Proof cont’d)

Appendix (C) : Minimal solutions (cont’d)

This is because if \((x_1, \ldots, x_m, y_1, \ldots, y_n) \geq (p \times q)\), then
\[
 w = \sum_{i \in I} (x_i - p(i)) u_i + \sum_{i \not\in I} (x_i + p(i)) u_i - \sum_{j \in J} (y_j - q(j)) v_j - \sum_{j \not\in J} (y_j + q(j)) v_j
\]
such that \( x_i - p(i) < x_i \) for some \( i \in I \) or \( y_j - q(j) < y_j \) for some \( j \in J \). By repeating the above computation, we obtain a positive solution of \((+1)\) such that \( x_i \leq p(i) \) for some \( i \in I \) or \( y_j \leq q(j) \) for some \( j \in J \). Let \( X = \{(k, \ell) \mid a \in I, k \leq p(i) \} \) and \( Y = \{(\ell, b) \mid b \in J, \ell \leq q(j) \}\). Then, the equation \((+1)\) has a positive solution if and only if there exists \((k, a) \in X \) such that \( w - ku_a = \sum_{i \in I - \{a\}} p(i) u_i - \sum_{j \in J} q(j) v_j \) \((+2)\) has a solution or \((\ell, b) \in Y \) such that \( w + \ell v_b = \sum_{i \in I} p(i) u_i - \sum_{j \in J - \{b\}} q(j) v_j \) \((+3)\) has a solution. Here “\((+2)\) has a solution” means the equation \((+2)\) has a positive solution or the solution is \( 0 \) (where \( k > 0 \) and the other \( p(i) \)’s are \( 0 \)). Similar to \((+3)\). By induction hypothesis, the question if \((+2)\) or \((+3)\) has a solution is decidable. Hence, since \( X, Y \) are finite, the question if \((+1)\) has a positive solution is decidable.

Next, we show our statement. According to the above observation, one can determine if there is a positive solution of \((+1)\). If there is no solution, the empty set is the answer. Otherwise, one can find a positive solution of \((+1)\). Since the number of vectors smaller than the solution is finite, one can find a minimal positive solution of \((+1)\), say \( s \). If there is another minimal positive solution of \((+1)\), say \( t \), then for some \( c, d \in \{ 1, \ldots, m + n \} \), \( t(c) < s(c) \) and \( t(d) > s(d) \). So, if \( 1 \leq c \leq m \), consider the equation \((+2)\) where \( k = c \) and \( u_a = t(c) \). (Proof cont’d)
Similarly, if $m < c \leq n$, consider the equation (\*3) where $\ell = c$ and $v_b = t(c)$. Since the number of the candidates for the pairs of such $(c, t(c))$ is finite, by induction hypothesis, one can compute the set $S_c$ of minimal positive solutions of (\*2) and the set $T_c$ of minimal positive solutions of (\*3). Let $f_{x,c}$ be the function from $\mathbb{N}^{m+n-1}$ to $\mathbb{N}^{m+n}$ such that $f_{x,c}(w) = (w(1), \ldots, w(c-1), x, w(c), \ldots, w(m+n-1))$. Then, $S = \min_>(\bigcup_{1 \leq c \leq m+n} \bigcup_{0 \leq x \leq s(c)} \{f_{x,c}(w) \mid w \in S_c \cup T_c\})$. Hence, we can compute the set of minimal positive solutions of (\*1). \qed
VIII. Equational tree automata
Tree automata (TA) revisited

\[ \text{TA} \left( F, Q, Q_{\text{fin}}, \Delta \right) \]

\( F \): signature

\( Q \): finite set of state symbols such that \( F \cap Q = \emptyset \)

\( Q_{\text{fin}} \): finite set \( Q_{\text{fin}} (\subseteq Q) \) of final states

\( \Delta \): finite set of transition rules with the following forms

- \( f(p_1, \ldots, p_n) \rightarrow q \) \hspace{1cm} [regular rule]
- \( p \rightarrow q \) \hspace{1cm} [epsilon rule]

if \( f \in F(n) \) \& \( p_1, \ldots, p_n, p, q \in Q \)

\( \text{regular} \) tree automaton (RTA) if \( \Delta \) does not contain an epsilon rule

\[ C(\text{RTA}_F) = C(\text{TA}_F) \]

Equational tree automata (ETA)

\[ \text{ETA} \mathcal{A} = \left( \mathcal{E}, Q, Q_{\text{fin}}, \Delta \right), \text{ denoted by } \mathcal{A}_\mathcal{E} \text{ (for } \mathcal{E} \text{ to be explicit)} \]

\( \mathcal{E} \): equational theory \( (F, E) \)

\( F \): signature

\( E \): finite set of equations

\( Q \): finite set of state symbols such that \( F \cap Q = \emptyset \)

\( Q_{\text{fin}} \): finite set \( Q_{\text{fin}} (\subseteq Q) \) of final states

\( \Delta \): finite set of transition rules

- \( \mathcal{A}_\mathcal{E} \) is regular if \( \Delta \) does not contain an epsilon rule

Proportionally equivalent notions ...

\( \text{TA \ & \ ETA} \propto \ GTRS \ & \ EGTRS \ (G: \text{ground}) \propto \ TRS \ & \ ETRS \)
Accepted trees

Given an ETA $A_E : (E, Q, Q_{\text{fin}}, \Delta)$

$ s \rightarrow_{A_E} t$ (move relation) \quad : \quad \exists \ l \rightarrow r \in \Delta, \ C[] \in C_{F \cup Q}$

\[ s =_E C[l] \land t =_E C[r] \]

$t$ is accepted by $A_E$ \quad : \quad \exists \ q \in Q_{\text{fin}} : \ t \rightarrow_{A_E} \cdots \rightarrow_{A_E} q$

tree language \quad : \quad \text{some subset of } T_F

tree language $L(A_E)$ \quad : \quad \text{set of trees accepted by } A_E

$L$ is $E$-regular tree language \quad : \quad \exists \text{regular } A_E : L = L(A_E)$

$E$ is linear equational theory \quad : \quad E$ consists of linear equations

$E$ is AC-theory \quad : \quad (\text{See page 7, seminar talk 4})

$E$ is $A$-theory \quad : \quad (\text{See page 8, seminar talk 4})

Example

Consider $A = (E, Q, Q_{\text{fin}}, \Delta)$ where $E = (F, E)$

\[
\begin{align*}
F & : 0 \ 1 \ \lor \ \land \ \neg \\
E & : \lor(x, y) = \lor(y, x) \lor(\lor(x, y), z) = \lor(x, \lor(y, z)) \\
& \land(x, y) = \land(y, x) \land(\land(x, y), z) = \land(x, \land(y, z)) \\
Q & : q_0 \ q_1 \\
Q_{\text{fin}} & : q_1 \\
\Delta & : 0 \rightarrow q_0 \ 1 \rightarrow q_1 \ \neg(q_0) \rightarrow q_1 \ \neg(q_1) \rightarrow q_0 \\
& \lor(q_0, q_0) \rightarrow q_0 \ \lor(q_0, q_1) \rightarrow q_1 \ \lor(q_1, q_0) \rightarrow q_1 \ \lor(q_1, q_1) \rightarrow q_1 \\
& \land(q_0, q_0) \rightarrow q_0 \ \land(q_0, q_1) \rightarrow q_0 \ \land(q_1, q_0) \rightarrow q_0 \ \land(q_1, q_1) \rightarrow q_1
\end{align*}
\]

We take $\neg(\land(\lor(1, 0), \land(1, 0)))$:

\[
\begin{align*}
\neg(\land(\lor(1, 0), \land(1, 0))) & \rightarrow_{A_E}^* \neg(\land(\lor(q_1, q_0), \land(q_1, q_0))) \\
& \rightarrow_{A_E} \neg(\land(q_1, q_0)) \\
& \rightarrow_{A_E} \neg(q_0) \\
& \rightarrow_{A_E} q_1 \quad \text{accepted by } A_E
\end{align*}
\]
Equationally equivalent trees

Let $\mathcal{A}_\mathcal{E} : \text{ETA} (\mathcal{E}, Q, Q_{\text{fin}}, \Delta)$

\[ s \equiv_\mathcal{E} t \quad \& \quad s : \text{accepted by } \mathcal{A}_\mathcal{E} \Rightarrow t : \text{accepted by } \mathcal{A}_\mathcal{E} \]

Proof

\[ \exists q \in Q_{\text{fin}} : \; s \rightarrow^{A_\mathcal{E}} s' \rightarrow^{A_\mathcal{E}} \cdots \rightarrow^{A_\mathcal{E}} q \quad (\text{by assumption}) \]

\[ \exists l \rightarrow r \in \Delta, \; C \in C_{\mathcal{F}_0 \mathcal{Q}} : \; s =_\mathcal{E} C[l] \; \& \; s' =_\mathcal{E} C[r] \]

\[ \Downarrow \quad s =_\mathcal{E} t \quad (\text{by assumption}) \]

\[ \exists l \rightarrow r \in \Delta, \; C \in C_{\mathcal{F}_0 \mathcal{Q}} : \; t =_\mathcal{E} C[l] \; \& \; s' =_\mathcal{E} C[r] \]

\[ t \rightarrow^{A_\mathcal{E}} s' \rightarrow^{A_\mathcal{E}} \cdots \rightarrow^{A_\mathcal{E}} q \]

Remark

– Regardless of $\mathcal{E}$, this property holds, i.e. the class $C(\text{ETA}_F)$ of tree languages is a sub-class of quotient sets of trees

– Regardless of $\mathcal{E}$, is $C(\text{ETA}_F)$ closed under Boolean operations?

Cf. $C(\text{TA}_F)$ is closed under Boolean operations

Closure properties of ETA (A-case)

Consider $\mathcal{A}_\mathcal{E} : \text{ETA}$ with $\mathcal{E} = (F, E)$ such that

\[ F = \{ f \} \cup F_0 \quad (F_0 : \text{set of constants}) \]

\[ E = \{ f(f(x, y), z) = f(x, f(y, z)) \} \]

then \( \exists \text{CFG } g \) such that

\[ \mathcal{L}(\mathcal{A}_\mathcal{E}) = \{ t \in T_F \mid g \text{ generates } \text{leaf}(t) \} \]

Corollary

If $\mathcal{E}$ is A-theory, the class of ETA is not closed under intersection or complement

Proof

Suppose, for leading to the contradiction, that this class of ETA over the signature $\{ f \} \cup F_0$ is closed under intersection. Then, there exists an ETA $\mathcal{C}_\mathcal{E}$ that accepts $\mathcal{L}(\mathcal{A}_\mathcal{E}) \cap \mathcal{L}(B_\mathcal{E})$. From the above observation, $\mathcal{L}(\mathcal{A}_\mathcal{E}) \cap \mathcal{L}(B_\mathcal{E}) = \{ t \in T_F \mid \exists \text{CFG } g_\mathcal{C} : g_\mathcal{A} \cap g_\mathcal{B} \text{ generates } \text{leaf}(t) \}$. However, since the class of CFG is not closed under intersection, there does not always exists such $g_\mathcal{C}$, that implies that $\mathcal{L}(\mathcal{A}_\mathcal{E}) \cap \mathcal{L}(B_\mathcal{E})$ is beyond this class. Similar for the proof of the complement. \( \square \)
Commutative words and AC-trees

Let $T = \{a, b\}$

\[
\begin{array}{c}
\text{aa abb} \\
\text{bab bba}
\end{array}
\]

Let $F = \{f, a, b\}$ (f is AC-symbol)

\[
\begin{array}{c}
\text{f(a,a) f(f(a,b),b)} \\
f(f(b,a),b) f(f(b,b),a)
\end{array}
\]

\[
\begin{array}{c}
(2) \\
(1)
\end{array}
\]

\[
\begin{array}{c}
(2) \\
(1)
\end{array}
\]

\[
\text{commutative CFL} \\
\text{(Presburger arithmetic)}
\]

\[
\text{AC-regular TL} \\
\text{(Presburger arithmetic + MSO)}
\]

Closure properties of regular ETA (AC-case)

Consider $A_\mathcal{E}$ : regular ETA with $\mathcal{E} = (F, E)$ such that

$F = \{f\} \cup F_0$

$E = \{f(f(x,y), z) = f(x, f(y,z)), f(x,y) = f(y,x)\}$

then $\exists$ NNVAS's $V_1, \ldots, V_n$ such that

$L(A_\mathcal{E}) = \{ t \in T_F \mid \#_{F_0}(\text{leaf}(t)) \in \llbracket V_1 \rrbracket \cup \ldots \cup \llbracket V_n \rrbracket \}$

* $\#_{F_0}$ is Parikh mapping from the alphabet $F_0^*$ to $\mathbb{N}^{|F_0|}$ (page 2, seminar talk 7)

Corollary

If $\mathcal{E}$ is AC-theory, the class of regular ETA is closed under Boolean operations

Proof for $\cap$

If $F = \{f\} \cup F_0$, this is an immediate consequence of the Boolean closedness of semi-linear sets (page 6, seminar talk 7). If $F = \{f\} \cup F'$ where $F'$ may contains non-constant symbols and $E = \{f(f(x,y), z) = f(x, f(y,z)), f(x,y) = f(y,x)\}$, let $A_\mathcal{E}$ and $B_\mathcal{E}$ be ETA such that $A = (\mathcal{E}, P, P_{\text{fin}}, \Delta_1)$ and $B = (\mathcal{E}, Q, Q_{\text{fin}}, \Delta_2)$ with the same $\mathcal{E}$. We suppose without loss of generality that $P \cap Q = \emptyset$. (Proof cont’d)
Proof for $\cap$ (cont’d)

Similar to the proof on page 11–12 in seminar talk 3, define $C_\varepsilon$ as follows.

$$P \times Q : = \{ (p, q) \mid p \in P, q \in Q \}$$

$$P_{\text{fin}} \times Q_{\text{fin}} : = \{ (p, q) \mid p \in P_{\text{fin}}, q \in Q_{\text{fin}} \}$$

$$\Delta_F := \left\{ (p_1, q_1), \ldots, (p_n, q_n) \rightarrow (p, q) \mid f(p_1, \ldots, p_n) \rightarrow p \in \Delta_1, f(q_1, \ldots, q_n) \rightarrow q \in \Delta_2, f \in F' \right\}$$

Moreover, for all $p \in P$ and $q \in Q$, define the inequations:

$$\text{ieq}_1(p, q) : = \langle p, q \rangle + \sum_{f(p_1, p_2) \rightarrow p \in \Delta_1, q \notin Q} x_{(p_1, q_1)} x_{(p_2, q_2)} \leq x_{(p, q)}$$

$$\text{ieq}_2(p, q) : = \langle p, q \rangle + \sum_{f(p_1, p_2) \rightarrow q \in \Delta_2, p \notin P} x_{(p_1, q_1)} x_{(p_2, q_2)} \leq x_{(p, q)}$$

Let $I_1 = \{ \text{ieq}_1(p, q) \mid p \in P, q \in Q \}$ and $I_2 = \{ \text{ieq}_2(p, q) \mid p \in P, q \in Q \}$. Each of $I_1, I_2$ contains language inequations with variables $V = \{ x_\ell \mid \ell \in P \times Q \}$. In the commutative Kleene algebra $K[V]$, there exists a unique solution for each of the systems $I_1$ and $I_2$ of inequations. Let $S_i (i \in \{1, 2\})$ be the solution, which contains equations in the form of $x_{(p, q)} = L$. The right-hand side $L$ is a language finitely represented by a commutative context-free grammar. Commutative context-free languages are isomorphic to semi-linear sets (Parikh’s theorem), and are effectively closed under intersection (page 6, seminar talk 7). So one can compute the set

$$S = \{ x_{(p, q)} = L \mid x_{(p, q)} = L_1 \in S_1, x_{(p, q)} = L_2 \in S_2 : L = L_1 \cap L_2 \}.$$  \hspace{1cm} \text{ (Proof cont’d) 10}

Proof for $\cap$ (cont’d)

To each $x_{(p, q)} = L$ in $S$, one can associate a context-free grammar $G_{(p, q)}$ in Chomsky normal form such that $c(L(G_{(p, q)})) = L$. One should remark that $G_{(p, q)}$ does not contain a production rule in the form of $\alpha \rightarrow \varepsilon$ (as $L$ does not contain $\varepsilon$). For each $p \in P$ and $q \in Q$, let $\Delta_{(p, q)}$ be the set of production rules of $G_{(p, q)}$ and let $\alpha_{(p, q)}$ be the starting symbol. Define

$$\Delta'_{(p, q)} = \{ \alpha \rightarrow \beta \gamma \mid \alpha \rightarrow \beta \gamma \in \Delta_{(p, q)} : \alpha \neq \alpha_{(p, q)} \} \cup \{ \langle p, q \rangle \rightarrow \beta \gamma \mid \alpha_{(p, q)} \rightarrow \beta \gamma \in \Delta_{(p, q)} \}. \hspace{1cm} \text{(*)}$$

Observe that $\langle p, q \rangle \rightarrow^* w$ and $w \in (P \cup Q)^*$ if and only if $w = \langle p, q \rangle$ or $w \in L(G_{(p, q)}).$

For each $p \in P$ and $q \in Q$, define

$$T_{(p, q)} = \{ f(\alpha, \beta) \rightarrow \gamma \mid \alpha \rightarrow \beta \in \Delta'_{(p, q)} \}. \hspace{1cm} \text{(**)}$$

From the above observation, for every $t \in T_{(f|_{(P \times Q)}}$ with $|t| \geq 3$ :

$$t \rightarrow^*_{T_{(p, q)}} \langle p, q \rangle \text{ if and only if } \pi_1(t) \rightarrow^*_{\Delta_1} p \& \pi_2(t) \rightarrow^*_{\Delta_2} q \text{ \hspace{1cm} (***)}$$

where $\pi_i (i \in \{1, 2\})$ is the projection $\pi_i((c_1, c_2)) = c_i$ ; $\pi_i(f(t_1, t_2)) = f(\pi_1(t_1), \pi_2(t_2))$.

Finally, let $\Delta_\varepsilon = \bigcup_{p \in P, q \in Q} T_{(p, q)}$. We take $\Delta_F \cup \Delta_\varepsilon$ as the set of transition rules of $C_\varepsilon$. From the above property (***) together with the argument ($s \rightarrow^*_{\Delta} \langle p, q \rangle$ iff $s \rightarrow^*_{\Delta} p \& s \rightarrow^*_{\Delta} q$) in page 12 in seminar talk 3, $L(C_\varepsilon) = L(A_\varepsilon) \cap L(B_\varepsilon).$ \hspace{1cm} $\square$

The closedness under union is obvious. For the proof idea of the complement, consult our paper [Proof of Lemma 5, Ohsaki Seki & Takai RTA2003].
Example

\[ A_{1E} : \quad a \to p \quad \Rightarrow \quad L_1 : \quad f(x) = a + b + x^2 \leq x \]
\[ b \to p \]
\[ f(p, p) \to p \]
\[ f \text{ is AC symbol} \]

\[ L_1 = f'(f(0))^* \cdot f(0) = (a + b)^* \cdot (a + b) \]
\[ = (a + b)^+ \]

\[ A_{2E} : \quad a \to q \quad \Rightarrow \quad L_2 : \quad g(y) = a + b \leq y \]
\[ b \to q \]

\[ L_2 = g'(g(0))^* \cdot g(0) = 1^* \cdot (a + b) \]
\[ = a + b \]

\[ L(A_{1E}) \cap L(A_{2E}) \quad \Leftrightarrow \quad c(L_1) \cap c(L_2) : \text{effectively computable} \]
\[ = \varepsilon(\{ a b \}) \]
\[ = \varepsilon(L(A)) \]
\[ = L(A_E) \]

Tree automata that count

Let \( L_\equiv : \) tree language that satisfies for all \( t \) in \( L_\equiv \),

\[ |t|_a = |t|_b \]

the numbers of occurrences of “a” and “b” are the same

\[ L_\equiv \] is accepted by AC-TA (ETA with the AC-theory of \( f \)) with the transition rules
\[ a \to q_a, \quad b \to q_b, \quad c \to q, \quad f(q_a, q_b) \to q, \quad f(q, q) \to q \quad (q : \text{final state}) \]

Note

The class of AC-TA logically subsumes the class of arithmetic in the right table (called Presburger arithmetic). A formula \( \delta \) in \( P \) is satisfiable iff there is an assignment \( \alpha \) to free variables in \( \delta \) such that the closed formula \( \delta\alpha \) is true. One can show that a formula \( \delta \) is satisfiable over \( \mathbb{N} \) iff there is an AC-regular tree language \( L_\delta \) over \( F = \{ f \} \cup F_0 \) such that \( f \) is an AC symbol and \( L_\delta \) is a model of \( \delta \). The previous example \( L_\equiv \) is a model of \( x_1 = x_2 \).

\[ P := C \]
\[ \quad | \quad P \lor P \]
\[ \quad | \quad \neg( P ) \]
\[ C := \exists x_i( P ) \]
\[ \quad | \quad \sum_{i \in I} a_ix_i = b \]
\[ (a_i, b \in \mathbb{Z}) \]
Commutation lemma

Let \( \mathcal{A}_E : \text{ETA} \)

\[ \mathcal{L}(\mathcal{A}_E) = \{ t \mid \exists s \in \mathcal{L}(A) : s \equiv_E t \} \quad (\mathcal{L}(\mathcal{A}_E) \text{ is } \equiv_E\text{-closure of } \mathcal{L}(A)) \]

if \( E \) is linear

**Proof**

The inclusion "\( \supseteq \)" is an immediate consequence of the result on page 5. For the reverse inclusion "\( \subseteq \)", we show that \( \rightarrow_A \cdot \leftrightarrow_E \subseteq \rightarrow_{\mathcal{A}} \cdot \rightarrow_{\mathcal{A}} \) (\( \rightarrow_{\mathcal{A}} \) is the smallest reflexive relation containing \( \rightarrow_A \)). Let \( \mathcal{A} = (E, Q, Q_{\text{fin}}, \Delta) \) and \( \mathcal{E} = (F, E) \). Suppose \( s \rightarrow_A t \) and \( t \leftrightarrow_E u \). Then there exists a transition rule \( l \rightarrow r \in \Delta \) such that \( s = C[l] \) and \( t = C[r] \), and there exists an equation \( l' = r' \in E \) such that (1) \( t = D[l' \sigma] \) and \( u = D[r' \sigma] \), or (2) \( t = D[l' \sigma] \) and \( u = D[r' \sigma] \). We consider (1) only below, but the same argument can be applied to (2). Since \( r \) is a state symbol and \( l' \) does not contain any state symbol, if \( l' = r' \) is applied above the position of \( r \) (at an ancestor position), \( \sigma \) contains a mapping \( x \mapsto D[r'] \) such that \( x \) is a variable in \( l' \). In this case, let \( \sigma' = (\sigma - \{ x \mapsto D[r'] \}) \cup \{ x \mapsto D[l'] \} \), then \( s = D[l' \sigma'] \). Because \( l' = r' \) is a linear equation, \( D[l' \sigma'] \rightarrow_E D[r' \sigma'] \) and \( D[r' \sigma'] \rightarrow \mathcal{A} u \). (If \( x \) does not appear in \( r' \), \( D[r' \sigma'] = u \)). If \( l' = r' \) is applied at a non-ancestor position of \( r \), obviously the commutation holds. Hence, the commutation of \( \rightarrow_A \) over \( \equiv_E \) holds.

\[ \square \]

Corollary

The class of AC-TA is closed under Boolean operations

**Proof**

Let \( \mathcal{A}_E \) be a (possibly non-regular) AC-TA. According to Commutation lemma, \( \mathcal{L}(\mathcal{A}_E) = \{ t \mid \exists s \in \mathcal{L}(A) : s \equiv_E t \} \). Since it is possible to construct DTA \( \mathcal{B} \) such that \( \mathcal{L}(A) = \mathcal{L}(B) \), one can obtain a regular ETA \( \mathcal{B}_E \) such that \( \mathcal{L}(B_E) = \mathcal{L}(A_E) \). Because the class of regular AC-TA is closed under Boolean operations (page 7), so is the class of AC-TA.

<table>
<thead>
<tr>
<th></th>
<th>AC-TA</th>
<th>A-TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>closed under ( \cup )</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>closed under ( \cap )</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>closed under ( \setminus )</td>
<td>✓</td>
<td>–</td>
</tr>
</tbody>
</table>

**Summary of closure properties**

Any super-/sub-class of A-TA (AC-TA) closed under Boolean operations?
### Decidability (AC-case)

The following problems are decidable for the class of AC-TA:

- \( t \in \mathcal{L}(A_E) \) ? \hspace{1cm} (membership problem)
- \( \mathcal{L}(A_E) = \emptyset \) ? \hspace{1cm} (emptiness problem)
- \( \mathcal{L}(A_E) = \mathcal{T}_F \) ? \hspace{1cm} (universality problem)
- \( \mathcal{L}(A_E) \subseteq \mathcal{L}(B_E) \) ? \hspace{1cm} (inclusion problem)
- \( \mathcal{L}(A_E) = \mathcal{L}(B_E) \) ? \hspace{1cm} (equivalence problem)

**Proof**

Similar to the proof of the decidability of TA, it suffices to show that the emptiness problem is decidable, because thanks to closedness under Boolean operations, the membership, universality, inclusion, equivalence problems are spacial cases of the emptiness problem. Let \( A_E \) be an AC-TA. \( (E \) is the AC-theory for some of the binary symbols.) According to the previous Commutation lemma, \( \mathcal{L}(A_E) = \{ t \mid \exists s \in \mathcal{L}(A) : s =_E t \} \). This implies that \( \mathcal{L}(A_E) = \emptyset \) if and only if \( \mathcal{L}(A) = \emptyset \). Since the emptiness problem is decidable for TA, so is for AC-TA.

The complexity of the emptiness problem of AC-TA coincides with TA (linear time). The membership problem is **NP-complete**. Other problems are at least \( 2^{2^n} \) \((c > 0)\).

### Decidability (A-case)

The following two problems are decidable for the class of A-TA:

- \( t \in \mathcal{L}(A_E) \) ? \hspace{1cm} (membership problem)
- \( \mathcal{L}(A_E) = \emptyset \) ? \hspace{1cm} (emptiness problem)

However, the other three problems are **undecidable**:

- \( \mathcal{L}(A_E) = \mathcal{T}_F \) ? \hspace{1cm} (universality problem)
- \( \mathcal{L}(A_E) \subseteq \mathcal{L}(B_E) \) ? \hspace{1cm} (inclusion problem)
- \( \mathcal{L}(A_E) = \mathcal{L}(B_E) \) ? \hspace{1cm} (equivalence problem)

**Proof**

The decidability of membership problem follows from the following observation: Given a tree \( t \) and an A-TA \( A_E \) over the signature \( F \), every reachable tree \( s \) from \( t \) satisfies \( |s| \leq |t| \). Since the number of trees that satisfies this condition is finite, it can be determined if \( t \) is reachable to some final state.

The decidability of emptiness problem is an immediate consequence of Commutation lemma (which implies that \( \mathcal{L}(A_E) = \emptyset \) if and only if \( \mathcal{L}(A) = \emptyset \) as the associativity axiom is a linear equation).
Proof (cont’d)

For undecidability of the universality, inclusion, equivalence problems, we use the reduction from the same problems of context-free grammar: Given two context-free grammar $G_1$ and $G_2$ over the alphabet $\Sigma$, define the A-TA $A_1$ and $B_2$ associated to $G_1$ and $G_2$, respectively. Without loss of generality, suppose $G_1$ and $G_2$ are in Chomsky normal form, and $L(G_i)$ ($i \in \{1, 2\}$) does not contain the empty word $\epsilon$. Then, every production rule of $G_i$ ($i \in \{1, 2\}$) is in the form of $\alpha \rightarrow \beta \gamma$ or $\alpha \rightarrow a$ ($\alpha, \beta, \gamma \in N_i$, $a \in T$, where $N_i$ is the set of non-terminals of $G_i$ and $T$ is the set of terminals). Let $F$ be the signature which contains $f$ (binary symbol) and constant symbols from $T$, and let $E$ be the set of associativity axiom of $f$. Define $A_1 = (E, Q_1, Q_{1\text{fin}}, \Delta_1)$ and $A_2 = (E, Q_2, Q_{2\text{fin}}, \Delta_2)$ as follows:

$Q_i : N_i$
$Q_{1\text{fin}} : \{ q_i \}$ where $q_i$ is a starting symbol of $G_i$
$\Delta_i : \{ f(\alpha, \beta) \rightarrow \gamma \mid \gamma \rightarrow \alpha \beta \in \Delta_i \} \cup \{ a \rightarrow \alpha \mid \alpha \rightarrow a \in \Delta_i \}$

Then the rest of the proof is straightforward, because for every word $w \in T^*$ and tree $t \in T_F$ such that leaf($t$) = $w$, $w \in L(G_i)$ if and only if $t \in L(A_{1\text{f}})$. Moreover, for every word $w \in T^* - \{\epsilon\}$, there exists a tree $t \in T_F$ such that leaf($t$) = $w$. (This is the proof of the reverse of the statement on page 6.) So, $L(G_1) \cup \{\epsilon\} = T^*$ if and only if $L(A_{1\text{f}}) = T_F$ (universality); $L(G_1) \subseteq L(G_2)$ if and only if $L(A_{1\text{f}}) \subseteq L(A_{2\text{f}})$ (inclusion); $L(G_1) = L(G_2)$ if and only if $L(A_{1\text{f}}) = L(A_{2\text{f}})$ (equivalence). Since these problems are undecidable for context-free grammar, so are for A-TA.

Related work

• Multitree automata
  (first appeared in 2003)

  D. Lugiez: Multitree automata that count, TCS 333, pp. 225–263, 2005

• Alternating two-way AC-tree automata
  (first appeared in 2003)


• Presburger tree automata


• Propositional tree automata * strictly more powerful than ETA

References

- Beyond Regularity: Equational Tree Automata for Associative and Commutative Theories (Ohsaki)
  15th CSL, LNCS 2142, pp. 539–553, 2001
- Decidability and Closure Properties of Equational Tree Languages (Ohsaki & Takai)
  13th RTA, LNCS 2378, pp. 114–128, 2002
- Recognizing Boolean Closed A-Tree Languages with Membership Conditional Rewriting Mechanism (Ohsaki & Seki & Takai)
  14th RTA, LNCS 2706, pp. 483–498, 2003
- ACTAS: A System Design for Associative and Commutative Tree Automata Theory (Ohsaki & Takai)
  5th RULE (2004), ENTCS 124, pp. 97–111, 2005
  Tool will be available at: http://staff.aist.go.jp/hitoshi.ohsaki/actas/

Exercise

1. Show that the AC-TA on page 13 accepts $L_\equiv$.
2. Show that the AC-closure of a regular tree language is not always a regular tree language. The AC-closure of $L$ is $\{ t \mid \exists t \in L: s =_E t \}$ where $E$ is the AC-theory. (Cf. Exercise 6,7, seminar talk 3)
3. Show that $L_\equiv$ is accepted by A-TA, i.e. let $E$ be the A-theory of $f$ with $F = \{ f, a, b, c \}$ and let $L_\equiv$ be the set of trees over $F$ satisfying $|t|_a = |t|_b$, show that $L_\equiv$ is accepted by A-TA. (Cf. Example on page 5, seminar talk 2)
4. If $F = \{ f, a \}$ with $ar(a) = 0$ and $ar(f) = 2$, the class $C(AC-TA_F)$ of AC-tree automata and the class $C(TA_F)$ of tree automata coincides.
5. Construct AC-TA $A_E$, $B_E$, $C_E$ over $F = \{ f \} \cup F_0$, each Parikh image of whose leaf languages is a model of (1) $\exists x_1, x_2 (x_1 = 1 \lor x_2 = 2)$, (2) $\forall x_1, x_2, x_3 (x_1 + x_2 = x_3)$, (3) $\forall x_1, x_2 (x_1 \geq 2 x_2)$.
6. Explain by showing an example why Commutation lemma (page 14) does not hold for a non-linear equational theory.
IX. Monotone tree automata
Monotone rules in tree automata

tree automaton \((F, Q, Q_{\text{fin}}, \Delta)\)

\(F\) : signature

\(Q\) : finite set of state symbols, such that \(F \cap Q = \emptyset\)

\(Q_{\text{fin}}\) : finite set of final states, such that \(Q_{\text{fin}} \subseteq Q\)

\(\Delta\) : finite set of transition rules:

\[f(\alpha_1, \ldots, \alpha_n) \rightarrow \beta_1\] [regular rule]

\[\alpha_1 \rightarrow \beta_1\] [epsilon rule]

\[f(\alpha_1, \ldots, \alpha_n) \rightarrow f(\beta_1, \ldots, \beta_n)\] [monotone rule]

with \(f \in F(n)\) & \(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n \in Q\)

Note

\(\text{Class}(\text{MTA}_F) = \text{Class}(\text{TA}_F)\) for any \(F\)

MTA & TA

For every monotone TA \(\mathcal{A}\), there effectively exists a regular TA \(\mathcal{B}\) over
the same signature equivalent to \(\mathcal{A}\), i.e. \(\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})\)

Proof

Let \(\mathcal{A} = (F, Q, Q_{\text{fin}}, \Delta)\) be a monotone TA. Define \(\mathcal{B} = (F, Q, Q_{\text{fin}}, \Delta')\), where

\(\Delta' : f(p_1, \ldots, p_n) \rightarrow q \quad \text{if} \quad f(p_1, \ldots, p_n) \rightarrow_{\Delta} q \quad \text{for some} \quad f \in F, p_1, \ldots, p_n, q \in Q\)

Observe that \(\Delta'\) is finite, because the number of trees whose height is at most 1 is
finite. We show \(\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B})\). For \(\subset\), let \(t = f(t_1, \ldots, t_n)\) in \(T_{F \cup Q}\) and \(t \rightarrow_{\Delta} q\) for
some \(q \in Q\). Then \(t \rightarrow_{\Delta'} f(p_1, \ldots, p_n) \rightarrow_{\Delta} q\). Here we suppose that in the derivation
\(t \rightarrow_{\Delta} f(p_1, \ldots, p_n)\), there is no transition step at the root position (though it may
happen after \(f(p_1, \ldots, p_n)\)). This implies \(t_i \rightarrow_{\Delta} p_i\) (\(1 \leq i \leq n\)). Let \(J \subseteq \{1, \ldots, n\}\)
such that for each \(j \in J, t_j \in Q\). Define \(f(u_1, \ldots, u_n)\) such that \(u_i = p_i\) if \(i \notin J; u_i = t_i\) if \(i \in J\). Then \(t \rightarrow_{\Delta'} f(u_1, \ldots, u_n) \rightarrow_{\Delta} f(p_1, \ldots, p_n) \rightarrow_{\Delta} q\). By induction
hypothesis, \(t_i \rightarrow_{\Delta} u_i\) if \(i \notin J\); otherwise, \(t_i = u_i\). Moreover, by definition of \(\Delta'\),
we have \(f(u_1, \ldots, u_n) \rightarrow q\) in \(\Delta'\). For the reverse \(\supset\), suppose \(t \rightarrow_{\Delta} q\). Since \(\mathcal{B}\)
is regular, if \(t\) is a state, then \(t \equiv q\). For induction step, suppose \(t = f(t_1, \ldots, t_n)\).
Then \(t \rightarrow_{\Delta} f(p_1, \ldots, p_n) \rightarrow_{\Delta} q\). Here the final step is performed by a single transition
step, and \(t_i \rightarrow_{\Delta} p_i\) (\(1 \leq i \leq n\)). Then, by induction hypothesis, \(t_i \rightarrow_{\Delta} p_i\) (\(1 \leq i \leq n\)).
Moreover, by construction, \(f(p_1, \ldots, p_n) \rightarrow_{\Delta} q \in \Delta'\) implies \(f(p_1, \ldots, p_n) \rightarrow_{\Delta} q\).
Monotone $\mathcal{E}$-tree automata ($M\mathcal{E}$-TA)

equational tree automaton $A = (\mathcal{E}, Q, Q_{\text{fin}}, \Delta)$, denoted $A_{\mathcal{E}}$

$\mathcal{E}$ : equational theory $(F, E)$

$F$ : signature

$E$ : finite set of equations

$Q$ : finite set of state symbols

$Q_{\text{fin}}$ : finite set of final states

$\Delta$ : finite set of transition rules

In particular,

$A_{\mathcal{E}}$ is monotone AC-tree automaton if $\mathcal{E}$ is AC-theory

$\text{Class}(M\mathcal{E}\text{-TA}) \supset \text{Class}(\mathcal{E}\text{-TA})$ for $\mathcal{E} = \text{AC}, \text{A}, ...$

Accepted trees

Given an ETA $A_{\mathcal{E}} : (\mathcal{E}, Q, Q_{\text{fin}}, \Delta)$

$s \xrightarrow{A_{\mathcal{E}}} t$ (move relation) \quad \rightarrow \exists l \rightarrow r \in \Delta, C[] \in C_{F \cup Q}:

s =_{\mathcal{E}} C[l] \ \& \ t =_{\mathcal{E}} C[r]

$t$ accepted by $A_{\mathcal{E}}$ \quad \rightarrow \exists q \in Q_{\text{fin}}: \ t \xrightarrow{A_{\mathcal{E}}} ... \xrightarrow{A_{\mathcal{E}}} q$

$\mathcal{E}$-monotone tree language $L$ \quad \rightarrow \exists$ monotone ETA $A_{\mathcal{E}} : L = L(A_{\mathcal{E}})$

$\mathcal{E}(L)$, called $=_{\mathcal{E}}$-closure of $L$ \quad $\{ s \mid \exists t \in L: s =_{\mathcal{E}} t \}$

Remark

$\mathcal{E}(L(A)) \neq L(A_{\mathcal{E}})$ for monotone AC-TA, while it holds for regular AC-TA:

E.g., consider $A_{\mathcal{E}}$ with

$\Delta : a \rightarrow q_1 \ b \rightarrow q_2 \ c \rightarrow q_3 \ f(q_2, q_3) \rightarrow f(q_4, q_3) \ f(q_1, q_4) \rightarrow q_5 \ f(q_5, q_3) \rightarrow q$

When q is final state and f is AC-symbol, $A_{\mathcal{E}}$ accepts $f(f(a, b), c)$. But $A$ accepts no tree, and thus $L(A) = \emptyset = \mathcal{E}(L(A)) \neq L(A_{\mathcal{E}})$. 
One can define a monotone AC-TA $A_\mathcal{E}$ over the minimal signature $F = \{ f, a, b, c \}$ such that $t \in \mathcal{L}(A_\mathcal{E})$ iff $|t|_a \times |t|_b \geq |t|_c$.

**Proof**

Define $A = (\mathcal{E}, Q, Q_\text{fin}, \Delta_\alpha \cup \Delta_\beta)$,

$\Delta_\alpha : a \to \alpha \quad b \to \alpha \quad f(\alpha, \alpha) \to \alpha$

$\Delta_\beta : a \to q_a \quad b \to q_b \quad c \to q_c \quad q_b \to q_1 \quad q_1 \to q_2 \quad q_2 \to \beta$

$f(q_2, q_b) \to q_1 \quad f(q_a, \beta) \to \beta$

$f(q_3, q_1) \to f(q_a, q_1)$

$f(q_a, q_2) \to f(q_a, q_2)$

$f(q_a, q_c) \to q_2$

$Q_\text{fin} : \alpha \quad \beta$

**Claim 1** $t \to^*_{A_\mathcal{E}} \alpha$ if $|t|_c = 0$ ; $t \to^*_{A_\mathcal{E}} \beta$ if $|t|_a \times |t|_b \geq |t|_c > 0$

**Proof (cont’d)**

We prove the reverse of the claim 1. In the previous figure, $q_5^c$ stands for the sequence of $q_a$ of the length $x$. So in the derivation $t \to^*_{A_\mathcal{E}} f(q_a, q_b, q_c)$, every constant $a$ (resp. $b$, $c$) in $t$ is replaced by $q_a$ ($q_b$, $q_c$). The remaining derivation in the previous figure is obtained as the composition of the two derivations

1. $f(q_a, q_1, q_b, q_c) \to^*_{A_\mathcal{E}} f(q_5^c, q_1)$
2. $f(q_a, q_1, q_2^c) \to^*_{A_\mathcal{E}} f(q_5^c, q_2)$

That means, first we obtain from $f(q_a^x, q_b^y, q_c^z)$:

$f(q_a, q_b^{y-1}, q_1, q_c^z) = f(q_a^x, q_1, (q_b, q_c^z), \ldots, (q_b, q_c^{y-m-1}), q_c^z)$

and then, $f(q_a, q_b^{y-1}, q_1, q_c^z) \to^*_{A_\mathcal{E}} f(q_a, q_2, q_b^{y-m-1})$. The above property (1) follows from the fact that by using the two rules $f(q_a, q_1) \to f(q_a, q_1)$ and $f(q_a, q_c) \to q_2$, we can obtain $f(q_a, q_1, q_b, q_c) \to^*_{A_\mathcal{E}} f(q_a, q_2, q_b^{y-m-1})$. Then, moreover, we apply $q_1 \to q_2$ and $f(q_2, q_2) \to f(q_3, q_2)$, and finally apply $f(q_2, q_b) \to q_1$. It turns out that $f(q_a, q_1, q_2, q_b) \to^*_{A_\mathcal{E}} f(q_a, q_2, q_b^{y-m-1})$. Hence, as we have seen, given a tree $t$ such that $|t|_a \times |t|_b \geq |t|_c$, then $t$ is accepted by $A_\mathcal{E}$. In the following we show the claim 1 (the reverse of the above statement), that is, $t \in \mathcal{L}(A_\mathcal{E})$ implies $|t|_a \times |t|_b \geq |t|_c$. (Proof cont’d)
As an easy observation, one can see that if \( t \rightarrow^{*_{\mathcal{A}_c}} \alpha \), then \(|t|_c = 0\). So it suffices to show that for every \( t \) in \( T_F \), if \( t \rightarrow^{*_{\mathcal{A}_c}} \beta \), then \(|t|_c > 0\) and \(|t|_a \times |t|_b \geq |t|_c\).

Define the following three mappings for trees in \( T_{F \cup Q} \):

\[
    f(t) = (|t|_a + |t|_{q_a} + |t|_{q_{a1}} + |t|_{q_{a2}}) \times (|t|_b + |t|_{q_b} + |t|_{q_1} + |t|_{q_2} + |t|_\beta) - (|t|_c + |t|_{q_c} + |t|_{q_2}) - (|t|_{q_2} + |t|_\beta) \times (|t|_a + |t|_{q_3})
\]

\[
    g(t) = |t|_{q_1} + |t|_{q_2} + |t|_\beta
\]

\[
    h(t) = (|t|_c + |t|_{q_c} - |t|_{q_{a2}})
\]

We claim that \( t \rightarrow^{*_{\mathcal{A}_c}} \beta \) implies \( f(t) \geq 0 \), \( g(t) \leq 1 \), \( h(t) \geq 0 \). By assumption, the rules \( a \rightarrow \alpha, b \rightarrow \alpha, f(\alpha, \alpha) \rightarrow \alpha \) in \( \Delta_\alpha \) can be ignored in the following discussion. We show the above loop invariant property, by induction on the number \( n \) of transition steps. If \( n = 0 \), then \( t \equiv \beta \), and thus \( f(t) = h(t) = 0 \) and \( g(t) = 1 \). For induction step \( n > 0 \), suppose \( t \rightarrow^{*_{\mathcal{A}_c}} \beta \). The proof proceeds by case analysis on the transition step \( t \rightarrow^{*_{\mathcal{A}_c}} t' \). At the transition performed by \( a \rightarrow q_a \) or \( b \rightarrow q_b, c \rightarrow q_c \), obviously the property holds. In case of the transition step by \( q_b \rightarrow q_1 \) (or \( q_1 \rightarrow q_2, q_2 \rightarrow \beta \)), we have \( g(t') = g(t) + 1 \). If \( g(t) = 1 \), it contradicts to \( t \rightarrow^{*_{\mathcal{A}_c}} \beta \), because \( g(t) \leq g(u) \) for every \( u \) in \( t \rightarrow^{*_{\mathcal{A}_c}} u \rightarrow^{*_{\mathcal{A}_c}} \beta \). Since \(|t|_{q_1} + |t|_{q_2} + |t|_\beta = 0\) in this case, we have \( f(t) = f(t') \) and \( h(t) = h(t') \).

(Proof cont’d)

In case of \( f(q_a, \beta) \rightarrow \beta \), \( f(t) - f(t') = (|t'|_a + |t'|_{q_a} + |t'|_{q_1} + |t'|_{q_2} + |t'|_\beta) - (|t'|_{q_2} + |t'|_\beta) = |t'|_a + |t'|_{q_a} + |t'|_{q_1} \geq 0 \), and we have \( g(t) = g(t') \) and \( h(t') = h(t') \). In case of \( f(q_a, q_c) \rightarrow q_{a2} \), \( h(t) - h(t') = -1 + 1 = 0 \), and \( f(t) = f(t') \) and \( g(t) = g(t') \). In case of \( f(q_b, q_1) \rightarrow q_1 \), we have \( f(t) - f(t') = (|t'|_a + |t'|_{q_a} + |t'|_{q_1} + |t'|_{q_2}) - (|t'|_{q_1} + |t'|_{q_2}) = |t'|_{q_1} + |t'|_{q_2} \geq 0 \), and \( g(t) = g(t') \) and \( h(t') = h(t') \). For monotone transition steps (2 cases), we look carefully. If \( f(q_a, q_1) \rightarrow f(q_{a1}, q_1) \) is applied, \( f(t) - f(t') = -(|t'|_{q_2} + |t'|_\beta) \) and \( g(t) = g(t') \). Since \( g(t') \leq 1 \) (induction hypothesis) and \(|t'|_{q_1} = 1 \), we have \(|t'|_{q_2} = |t'|_{q_1} = 0 \), and thus \( f(t) - f(t') = 0 \). On the other hand, \( h(t') = h(t) - 1 \). If \( h(t) \leq 0 \), then extra \( q_{a1} \)'s in \( t' \) remain in the following derivation, that contradicts to \( t \rightarrow^{*_{\mathcal{A}_c}} \beta \). Thus, \( h(t) > 0 \), and then \( h(t') \geq 0 \). In case of \( f(q_{a2}, q_2) \rightarrow f(q_a, q_2) \), \( f(t) - f(t') = -1 - (|t'|_{q_2} + |t'|_\beta) \times (-1) \). Similar to the previous case, since \(|t'|_{q_2} = 1 \), we have \(|t'|_\beta = 0 \) by induction hypothesis. Thus, \( f(t) - f(t') = -1 + 1 = 0 \). Besides, \( g(t) = g(t') \) and \( h(t') = h(t) \). Hence, if \( t \in T_F \) and \( t \rightarrow^{*_{\mathcal{A}_c}} \beta \), then \(|t|_a \times |t|_b \geq |t|_c \).

(Proof cont’d)

Remark

One can show that \((\Delta_1 \cup \Delta_2) / \mathcal{E}\) is terminating. By taking \( \rightarrow^{*_{\mathcal{A}_c}} \) as a well-founded order, we then can show by induction on this order that “if \( f(t) \geq 0 \), \( g(t) \leq 1 \), \( h(t) \geq 0 \), there exists a derivation \( t \rightarrow^{*_{\mathcal{A}_c}} C[q_{a_2}] \) for some \( C \in C(t,a_1) \),” provided that \(|t'|_{q_2} + |t'|_{q_3} + |t|_{q_1} + |t|_{q_2} \geq 1 \) and \(|t|_{q_1} = 0 \). This also yields the proof of the claim 1.
One can modify $A_ε$ to $B_ε$ such that $t \in L(B_ε)$ iff $|t|_a \times |t|_b > |t|_c$

**Proof**

Define $B_ε = (ε, Q \cup \{ p_a, p_b, q_{b1}, q_{c1}, q_{c2}, q_{f1}, q_{f2}, q_{f3} \}, \{ p_r, q_{t3} \}, \Delta'_1 \cup \Delta'_2 \cup \Delta'_3)$, where

\[ \Delta'_1 = \{ a \rightarrow p_a, b \rightarrow p_b, f(p_a, p_b) \rightarrow p_r, f(p_a, q_{t3}) \rightarrow q_{f1}, f(p_b, q_{t3}) \rightarrow q_{f2} \} \]

\[ \Delta'_2 = \{ f(q_{a1}, q_{c1}) \rightarrow q_{c1}, f(q_{a1}, q_{c2}) \rightarrow q_{c2}, f(q, q_{t3}) \rightarrow q_{f2} \} \]

\[ \Delta'_3 = \{ f(q_{a1}, q_{c1}) \rightarrow q_{c1}, f(q_{a1}, q_{c2}) \rightarrow q_{c2}, f(q_{b1}, q_{f1}) \rightarrow q_{f1}, f(q_{b1}, q_{f2}) \rightarrow q_{f2} \} \]

If $|t|_c = 0$, then $t$ contains $a$ and $b$ iff $t$ is accepted as $t \rightarrow^{*}_{B_ε} p_r$. If $|t| > 0$, we claim that $t \rightarrow^{*}_{B_ε} q_{f2}$ iff $|t|_a \times |t|_b > |t|_c$. Let $t = f(a^2, b^y, c^z)$ and $z = x \times m + (n - 1)$ such that $0 \leq m < y$ and $0 < n \leq x$, i.e. $x \times y > z$, then

\[ t \rightarrow^{*}_{B_ε} f(q_{a1}, q_{b1} y^{-1}, q_{c1}, q_{f1}) = f(q_{a1}, q_{b1}, q_{c1}, \ldots, q_{c1}, q_{b1}, q_{c1} y^{-m-1}, q_{c1} n^{-1}) \]

If $x = 1$, then $x \times y > z$ iff $y > z$. This implies $t \rightarrow^{*}_{B_ε} f(q_{a1}, q_{b1}, q_{c1}, q_{f1}) = f(q_{a1}, q_{b1} y^{-1}, q_{c1}, q_{f1})$. Thus, $f(q_{a1}, q_{b1}, q_{c1}, q_{f1}) \rightarrow^{*}_{B_ε} f(q_{b1}, q_{f1}) \rightarrow^{*}_{B_ε} q_{f1}$. If $x \geq 2$, from the property (1) in the previous proof, we can erase at most $(x \times m)$ states of $q_c$ and $m$ states of $q_{b1}$ from $f(q_{a1}, q_{b1}, q_{c1}, q_{f1})$. Moreover, by using the rules $f(q_{a1}, q_{c1}) \rightarrow q_{c1}$ and $f(q_{a1}, q_{c1}) \rightarrow q_{f1}$, we have $f(q_{a1}, q_{b1} y^{-m-1}, q_{c1} n^{-1}) \rightarrow^{*}_{B_ε} f(q_{a1}, q_{b1} y^{-m-1}, q_{c1} n^{-1}) \rightarrow^{*}_{B_ε} f(q_{b1}, q_{f1}) \rightarrow^{*}_{B_ε} q_{f1}$.

**Arithmetic constraints**

1. **ψ** is exponential Diophantine

   if **ψ** is in $D$.

   $D := A$

   $|\neg(D)$

2. **ψ** is Diophantine

   if **ψ** contains no $(\ast 2)$-formula.

   $|D \lor D$

   $|D \land D$

3. **ψ** is linear (or Presburger formula)

   if **ψ** contains no $(\ast 1)$-, $(\ast 2)$-formula.

   $A := \exists x_i(D)$

   $|\sum_{i \in I} a_i x_i > b$

4. **ψ** is monotone

   if negative sub-formula of **ψ**

   contains no $(\ast 1)$-, $(\ast 2)$-formula.

   $|x_i x_j \geq x_k \cdots (\ast 1)$

   $|x_i x_j \geq x_k \cdots (\ast 2)$

**Note**

linear $\subset$ monotone (exponential) Diophantine $\subset$ (exponential) Diophantine
Satisfiability

A formula $\psi$ is **satisfiable** if $\exists \theta$ (assignment over $\mathbb{N}$ to free variables in $\psi$) such that a closed formula $\psi \theta$ is true.

**Example**

- $x^3 \geq y$ : monotone Diophantine & satisfiable
  
  \[
  \vdash \Leftrightarrow \exists z \ (x \times z \geq y \land x \times x \geq z)
  \]

- $x^x \geq y$ : monotone exponential Diophantine & satisfiable
  
  \[
  \vdash \Leftrightarrow \exists z \ (x^z \geq y \land x^x \geq z)
  \]

- $x^2 = y$ : Diophantine & not monotone & satisfiable
  
  \[
  \vdash \Leftrightarrow (x^2 \geq y) \land \exists z \ (-(x^2 \geq z) \land z = y + 1)
  \]

- $x \geq y \land y \geq x + 1$ : monotone & not satisfiable

**Solution set**

Let $\psi$ : arithmetic formula with free variables $x_1, \ldots, x_n$

- the **solution set** $[[\psi]]_\mathbb{N}$ over the domain $\mathbb{N}$ is the set of vectors

  \[
  [[\psi]]_\mathbb{N} = \left\{ \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \mid \exists c_1, \ldots, c_n \in \mathbb{N} : \psi [x_1 \mapsto c_1, \ldots, x_n \mapsto c_n] \right\}
  \]

- $i$-th **projection** $pr_i$ ($1 \leq i \leq n$) is a mapping

  \[
  pr_i(S) = \left\{ \begin{pmatrix} c_1 \\ \vdots \\ c_i \\ c_{i+1} \\ \vdots \\ c_n \end{pmatrix} \mid \exists v \in S : v = \begin{pmatrix} c_1 \\ \vdots \\ c_i \\ c_{i+1} \\ \vdots \\ c_n \end{pmatrix} \right\}
  \]

  Cf. cylindrification $cy_i$ ($1 \leq i \leq n+1$)

**Note**

$[[\exists x_i(\psi)]] = pr_i([[\psi]])$
Diophantine formulas

For the class of polynomials with integer coefficients, the question if
\[ P(x_1, \ldots, x_n) = 0 \] is \( \emptyset \) is undecidable [1]. Original Hilbert’s 10th problem is the question if \( P(x_1, \ldots, x_n) = 0 \) is \( \emptyset \), and the undecidability result can be extended to \( \mathbb{N} \) ([2]).

Corollary

Satisfiability of (exponential) Diophantine formulas is undecidable

Proof

Given a polynomial \( P(x_1, \ldots, x_n) \) with integer coefficients, \( P(x_1, \ldots, x_n) = 0 \) is a Diophantine formula. In fact, \( P(x_1, \ldots, x_n) \) is represented as \( \sum_{i \in I} c_i \times m_i(x_1, \ldots, x_n) = y_i \) for some constant \( c_i \in \mathbb{Z} \) and monomial \( m_i \) \( i \in I \). Then \( P(x_1, \ldots, x_n) = 0 \) iff \( \sum_{i \in I} y_i = 0 \land \bigwedge_{i \in I}(c_i \times m_i(x_1, \ldots, x_n) = y_i) \). Note that \( a = b \) iff \( (a \geq b) \land \neg(a \geq b + 1) \). Obviously, \( [P(x_1, \ldots, x_n) = 0] \neq \emptyset \) iff \( P(x_1, \ldots, x_n) = 0 \) is satisfiable. However, according to the above undecidability result of Hilbert’s 10th problem, satisfiability of \( P(x_1, \ldots, x_n) = 0 \) is undecidable.


Presburger formulas

Satisfiability of Presburger formulas is decidable

Proof

Define our interpretation of formulas \( \psi \) with a (finite) set \( V \) of free variables: \( [\psi_1 \lor \psi_2]_N = [\psi_1]_N \cup [\psi_2]_N \); \( \neg(\psi_1)_N = ([\psi_1]_N)^C \); \( \exists x_k(\psi_1)_N = \text{pr}_k([\psi_1]_N) \). If \( V_1 \neq V_2 \), cylindrify elements to a solution set at corresponding variable positions. For instance, if \( \psi_1 \) has free variables \( x_1, x_2 \) and \( \psi_2 \) has a free variable \( x_1 \), \( [\psi_1]_N \cup [\psi_2]_N \) means \( [\psi_1]_N \cup \{ (v,c_2) \mid v \in [\psi_2]_N, c_2 \in \mathbb{N} \} \). Note that if \( R(x_1, \ldots, x_n) \) is a linear polynomial with integer coefficients, then \( [R(x_1, \ldots, x_n) \geq 0]_N \) is a semi-linear subset of \( \mathbb{N}^n \). Moreover, the semi-linearity is closed under Boolean operations (page 6, seminar talk 7) and closed under projection and cylindrification (page 21, seminar talk 7). Thus, if \( \psi \) is a Presburger formula, then \( [\psi]_N \) is semi-linear. Since \( [\psi]_N = \emptyset \) is decidable for the class of semi-linear sets, so is the satisfiability of \( \psi \).

Corollary [ Ginsburg & Spanier ]

A subset \( S \) of \( \mathbb{N}^n \) is semi-linear iff \( \exists \) Presburger formula \( \psi \) such that \( S = [\psi]_N \) with \( n \) free variables. \( [\psi]_N \) is called a Presburger set

Monotone Diophantine formulas

1. If the class of monotone AC-TA is effectively closed under Boolean operations, the emptiness problem is undecidable

   * “Effective closedness” guarantees not only the existence of effective procedure for the desired computation, but the procedure (the proof for closedness of operations) to be constructable

2. If the class of monotone AC-TA is effectively closed under $\cup$, $\cap$ and the emptiness problem is decidable, then satisfiability of monotone Diophantine formulas is decidable

Proof of 1

One can show that for every formula $P(x_1,\ldots,x_m) = 0$ with integer polynomial $P(x_1,\ldots,x_m)$, one can find an equivalent formula $\exists \vec{x}_n(\psi)$ for some $\psi$ in $S_1$:

$S_1 ::= x \times y \geq z \mid C \text{ (Presburger formula)} \mid S_1 \lor S_2 \mid S_1 \land S_2 \mid (S_1)^c$

According to the previous result (page 7), there effectively exists a monotone AC-TA that represents $x \times y \geq z$. Moreover, every Presburger formula can also be represented by monotone AC=TA (even by regular AC-TA). So if the class of monotone AC-TA is closed under Boolean operations, then one can construct a monotone AC-TA $A_{E}$ such that $\Psi(A_{E}) = [-\vec{0}]$. Note that there is no tree whose Parikh image is $\vec{0}$ (the vector whose elements are all zero). (Proof cont’d) 17

Proof of 1 (cont’d)

We should notice that $\exists \vec{x}_n(\psi)$ is satisfiable iff $L(A_{E}) \neq \emptyset$ or $\vec{0}$ is a solution of $\psi$. For a given polynomial $P(x_1,\ldots,x_m)$, (1) the question if $\vec{0} \in [-P(x_1,\ldots,x_m) = 0]_N$ is decidable, but (2) the question if $[-P(x_1,\ldots,x_m) = 0]_N = \emptyset$ is not decidable. Hence, if the class of monotone AC-TA is closed under all Boolean operations, then the question if $L(A_{E}) \neq \emptyset$ is undecidable (as the question if $\exists \vec{x}_n(\psi)$ is satisfiable is undecidable due to (2)).

Proof of 2

Similar to the previous proof. In this case, we observe that for every monotone Diophantine formula, one can find an equivalent formula $\exists \vec{x}_n(\psi)$ for some $\psi$ in $S_2$:

$S_2 ::= x \times y \geq z \mid C \text{ (Presburger formula)} \mid S_2 \lor S_2 \mid S_2 \land S_2$

For every non-linear inequality $x \times y \geq z$, there effectively exists a monotone AC-TA that represents $[-x \times y \geq z]_N - \vec{0}$. Moreover, every Presburger formula can also be represented by monotone AC=TA. So if the class of monotone AC-TA is closed under union and intersection, then one can construct a monotone AC-TA $B_{E}$ such that $\Psi(B_{E}) = [-\psi] - \vec{0}$. Thus, $\exists \vec{x}_n(\psi)$ is satisfiable iff $L(B_{E}) \neq \emptyset$ or $\vec{0}$ is a solution of $\psi$. Since the question if $\vec{0}$ is a solution of $\psi$ is decidable, the decidability of $L(B_{E}) = \emptyset$ implies the decidability of the satisfiability of $\exists \vec{x}_n(\psi)$. 18
The class of monotone AC-TA is effectively closed under union and intersection.

Proof for $\cup$

Obvious. Suppose $E = (F, E)$ is an AC-theory and $A_1 = (E, P, \mathcal{P})$ and $A_2 = (E, Q, \mathcal{Q})$ are monotone AC-TA whose sets $P, Q$ of state symbols are pairwise distinct. Define $B = (E, P \cup Q, \mathcal{P} \cup \mathcal{Q})$, then by construction, $B$ accepts a tree $t$ in $T_F$ if and only if $A_1$ or $A_2$ accepts $t$.

Proof for $\cap$

Recall the proof for the closure properties of CSG (pages 14-15, seminar talk 2). Let $A_1 = (E, P, \mathcal{P})$ and $A_2 = (E, Q, \mathcal{Q})$ be monotone AC-TA with the same AC-theory $E = (F, E)$, and denote $F_{AC}$ for the set of AC symbols. Define a monotone AC-TA $B = (E, (P \cup \{\emptyset\}) \times Q, \mathcal{P} \cup \mathcal{Q})$ where

$\Delta: f((p_1, q_1), \ldots, (p_n, q_n)) \rightarrow (p, q)$ if $\exists p_1, \ldots, p_n, p \in P, q_1, \ldots, q_n, q \in Q, f \in F - F_{AC}$:

$f(p_1, \ldots, p_n) \rightarrow^*_{A_1} p \land f(q_1, \ldots, q_n) \rightarrow^*_{A_2} q$

and

(Proof cont’d) 19

Proof for $\cap$ (cont’d)

$\Delta_{AC} = \bigcup_{f \in F_{AC}} \Delta_f$ such that for every AC symbol $f \in F_{AC}$,

$\Delta_f: f((p_1, q_1), (p_2, q_2)) \rightarrow f((p_1, q_1), (\emptyset, q_2))$ if $\exists f(p_1, p_2) \rightarrow p \in \Delta_1, q_1, q_2 \in Q$

$f((p_1, q_1), (p_2, q_2)) \rightarrow f((p_3, q_1), (p_4, q_2))$ if $\exists f(p_1, q_2) \rightarrow f(p_3, p_4) \in \Delta_1, q_1, q_2 \in Q$

$f((\emptyset, q_1), (p_2, q_2)) \rightarrow (p_2, q)$ if $\exists f(q_1, q_2) \rightarrow q \in \Delta_2, p_2 \in P$

$f((\emptyset, q_1), (p_2, q_2)) \rightarrow f((\emptyset, q_3), (p_2, q_4))$ if $\exists f(q_1, q_2) \rightarrow f(q_3, q_4) \in \Delta_2, p_2 \in P$

$\langle p_1, q_1 \rangle \rightarrow (p_2, q_1)$ if $\exists p_1 \rightarrow p_2 \in \Delta_1, q_1 \in Q$

$\langle p_1, q_1 \rangle \rightarrow (p_1, q_2)$ if $\exists q_1 \rightarrow q_2 \in \Delta_2, p_1 \in P$

$\langle p_1, q_1 \rangle \rightarrow (\emptyset, q_2)$ if $\exists p_1 \in P, q_1, q_2 \in Q$

$\langle p_1, q_1 \rangle \rightarrow (p_1, q_2)$ if $\exists p_1 \in P, q_1, q_2 \in Q$

Then the monotone AC-TA $B_E$ simulates $A_1$ and $A_2$ such that for every tree $t \in T_F$, $t \rightarrow^*_{E} \langle p, q \rangle$ iff $t \rightarrow^*_{A_1} p$ and $t \rightarrow^*_{A_2} q$. The above construction is based on the proof of [Theorem 3, Ohsaki 2001]. Observe that the same proof argument can be applied to the A-case. That means, by using the same components $\Delta \cup \Delta_{AC}$ and $(P \cup \{\emptyset\}) \times Q$ but an A-theory $E$, one can show that the class of monotone A-TA is closed under intersection. \Box

H. Ohsaki: Beyond regularity: Equational Tree Automata for Associative and Commutative Theories, 15th CSL, LNCS 2142, pp.539–553, 2001
**Emptiness problem**

The emptiness problem for monotone AC-TA is decidable

**Proof**

Given a monotone AC-TA $A_\epsilon = (\epsilon, P, P_{\text{fin}}, \Delta_1)$ with $\epsilon = (F,E)$, we define an AC-TA $B_\epsilon = (\epsilon, Q, Q_{\text{fin}}, \Delta_2)$ such that, by letting $q$ be a fresh symbol, $Q = Q_{\text{fin}} = \{q\}$ and $\Delta_2 = \{ f(q, \ldots, q) \to q \mid f \in F \}$. Obviously, $B_\epsilon$ accepts all trees over $F$. Moreover, $t \in L(B_\epsilon)$ iff $q \to^{\Delta_2 \cdot \epsilon} t$ where $\Delta_2^{-1} = \{ s \to t \mid t \to s \in \Delta_2 \}$. Let $\Delta_3 = \Delta_1 \cup \Delta_2^{-1}$, then one can show that for every $t \in T_F$ and $p \in P$, $t \to^{\Delta_1 / \epsilon} p$ iff $q \to^{\Delta_3 / \epsilon} t \to^{\Delta_3 / \epsilon} p$. It suffices to show that (1) $\to^{\Delta_1 / \epsilon} \to^{\Delta_2^{-1} / \epsilon} \subseteq \to^{\Delta_2^{-1} / \epsilon} \to^{\Delta_1 / \epsilon}$, (2) $q \to^{\Delta_3 / \epsilon} t \to^{\Delta_3 / \epsilon} p$ implies $q \to^{\Delta_1 / \epsilon} t \to^{\Delta_2^{-1} / \epsilon} p$. One should note that there exists a final state $p \in P_{\text{fin}}$ such that $q \to^{\Delta_3 / \epsilon} p$ iff $L(A_\epsilon) \neq \emptyset$. Since $P_{\text{fin}}$ is finite, there are only finitely many test cases of $q \to^{\Delta_3 / \epsilon} p$. Because

the reachability problem for AC-GTRS is decidable (Mayer & Rusinowitch 1998), the question if $L(A_\epsilon) \neq \emptyset$ is decidable.

**Corollary**

Satisfiability of monotone Diophantine formulas is decidable

**Corollary**

The class of monotone AC-TA is not closed under complement

**Proof**

For leading to the contradiction that there exists an effective procedure to solve $[P(x_1, \ldots, x_n) = 0]_{\mathbb{N}}$, we take the hypothesis that for the signature $F = \{ f, a, b, c \}$,

$\mathcal{H} \colon \{ t \mid |t|_a \times |t|_b \leq |t|_c \}$ is an AC-monotone tree language.

Given a polynomial $P(x_1, \ldots, x_n)$ with integer coefficients, $P(x_1, \ldots, x_n) = 0$ is represented as the conjunction of polynomial equations in the forms of $x_i = a$ ($a \in \mathbb{N}$), $x_i + x_j = x_k$, $x_i \times x_j = x_k$ ($k \neq i, j$). Observe that the equivalent polynomial equations in the above forms may require additional variables $y_1, \ldots, y_m$ ($m \geq 0$). Let $F = \{ f \} \cup \{ a_1, \ldots, a_n, b_1, \ldots, b_m \}$. Since the class of AC-monotone tree languages are closed under intersection, an effective procedure can be defined that takes $P(x_1, \ldots, x_n) = 0$ as the input and returns a monotone AC-TA $A_{P_\epsilon}$ over $F$ such that $P(x_1, \ldots, x_n) = 0$ is satisfiable iff $L(A_{P_\epsilon}) \neq \emptyset$. However, from the undecidability of Hilbert’s 10th problem, such a procedure does not exist, and thus $\mathcal{H}$ does not hold. Therefore, the complement of $\{ t \mid |t|_a \times |t|_b > |t|_c \}$ is not AC-monotone.

**Note**

$C(\text{ETA}_F) \not\subseteq C(\text{META}_F)$
**The inclusion problem for the class of monotone AC-TA is undecidable**

**Proof**

According to the previous proof, given a polynomial equation $P(x_1, \ldots, x_n) = 0$, it can be represented as the conjunction of polynomial equations in the forms of $x_i = a$ ($a \in \mathbb{N}$), $x_i + x_j = x_k$, $x_i \times x_j = x_k$ ($k \neq i, j$), then suppose $P(x_1, \ldots, x_n) = \land_{1 \leq i < \ell} p_i(x_1, \ldots, x_n, y_1, \ldots, y_m) = q$, such that $q_i \in \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ or $q_i \in \mathbb{N}$. Let $P(x_1, \ldots, x_n) \equiv \bigwedge_{1 \leq i \leq \ell} p_i(x_1, \ldots, x_n, y_1, \ldots, y_m) \geq q_i$ where $(\vec{x}_n, \vec{y}_m)$ stands for $(x_1, \ldots, x_n, y_1, \ldots, y_m)$.

Then $P(x_1, \ldots, x_n)$ is satisfiable iff $P_\succ(\vec{x}_n, \vec{y}_m)$ is satisfiable and for all $1 \leq i \leq \ell$, $Q_i(\vec{x}_n, \vec{y}_m)$ is not satisfiable, because: $[[P(x_1, \ldots, x_n) = 0]]_\mathbb{N} \neq \emptyset$ iff $[[P_\succ(\vec{x}_n, \vec{y}_m)]] \notin \bigcup_{1 \leq i \leq \ell} [[Q_i(\vec{x}_n, \vec{y}_m)]]$. Since $[[P_\succ(\vec{x}_n, \vec{y}_m)]]$ and $[[Q_i(\vec{x}_n, \vec{y}_m)]]$ ($1 \leq i \leq \ell$) are AC-monotone languages and the question if $[[P(x_1, \ldots, x_n) = 0]]_\mathbb{N} \neq \emptyset$ is undecidable (Hilbert’s 10th problem), the inclusion problem for monotone AC-TA is undecidable.

---

**Open questions**

1. For monotone AC-TA, the question if $L(A_e) = T_F$ is decidable? *(universality problem)* *open since 2002 (RTA open problem §101)*

   The universality problem is an instance of the inclusion problem, and the inclusion problem is undecidable. The universality problem is reformulated as $(L(A_e))^C \cap T_F = \emptyset$, but since the class of AC-monotone tree languages is not closed under complement, $(L(A_e))^C$ may not be AC-monotone, though the emptiness problem for monotone AC-TA is decidable and the class of AC-monotone tree languages is closed under intersection.

2. Is there any arithmetic logic that captures monotone AC-TA? *(complete logical characterization)*

   Recently we showed that a sub-class of monotone (exponential) Diophantine formulas is definable by monotone AC-TA (Theorem 3 in [1]). However, we still do not know exactly which class of the arithmetic can be the counterpart of monotone AC-TA. (Cf. regular AC-TA & Presburger MSO)

---

Open questions (cont’d)

3. The class of monotone AC-TA is closed under projection?

As explained at the beginning, Petri-nets are a special class of ground AC-TRS’s whose signature is an AC symbol and constants (flat signature). One can easily observe that by definition, every AC-monotone tree language is Petri-net definable. So the interesting question is the reverse: every Petri-net definable tree language is AC-monotone or not? A sufficient condition for the positive answer to this question is that the class of AC-monotone tree languages is closed under projection.

4. One can decide whether a given monotone exponential Diophantine formula is valid?

Linearly bounded projection eliminates a certain type of $\exists$-quantifiers in the formulas, which means that a sub-class of monotone exponential Diophantine formulas is monotone AC-TA definable. In the paper we remain the question whether every monotone exponential Diophantine formula can be transformed to a monotone AC-TA definable formula.

Exercise

1. Show that why the class of monotone AC-TA properly subsumes the class of regular AC-TA.

2. Construct monotone AC-TA over the signature $\{f, a, b, c\}$ with AC symbol $f$, whose Parikh’s image is:

   (1) $[2x + y = z]_N - \bar{0}$
   (2) $[x^2 \geq y]_N - \bar{0}$
   (3) $[2^x \geq y]_N - \bar{0}$

   (*) An arithmetic formula $\psi$ is monotone AC-TA definable iff there effectively exists monotone AC-TA $A_\epsilon$ such that $\Psi(L(A_\epsilon)) = [\psi]_N - \bar{0}$.

3. [Linearly bounded formulas] Show that if an arithmetic formula $\psi$ (not restricted to Diophantine formulas) is monotone AC-TA definable, then so is $\exists x (x \leq y \land \psi)$.

Refer [1] for Exercise 2(3) and 3.

Appendix (A) : Monotone Exponential Diophantine Formulas

Formulas in $D$ are monotone exponential Diophantine formulas:

$$\begin{align*}
D := x_i x_j & \geq x_k \mid x_i^x & \geq x_k \mid P \\
& \mid \exists x_i (D) \mid D \lor D \mid D \land D
\end{align*}$$

$$\begin{align*}
P := x_i + x_j & = x_k \mid x_i = c \ (c \in \mathbb{N}) \\
& \mid \exists x_i (P) \mid \forall x_i (P) \mid P \lor P \mid P \land P
\end{align*}$$

The class of formulas $\exists \vec{x}_n (\psi)$ where $\psi \in D'$ coincides with $D$:

$$D' := E(\vec{x}_n) \geq L(\vec{x}_n) \mid P \mid D' \lor D' \mid D' \land D'$$

where $E(\vec{x}_n)$ is an exponential with non-negative integer coefficients, and $L(\vec{x}_n)$ is a linear polynomial with integer coefficients.

Theorem [Kobayashi & Ohsaki RTA’08]

$E(\vec{x}_n) \geq L(\vec{x}_n)$ is definable by monotone AC-TA.

As an immediate consequence, the class $D'$ is definable by monotone AC-TA.
Appendix (B) : Summary

Solution sets are

- closed under Boolean operations:
  - Presburger arithmetic
  - (exponential) Diophantine arithmetic

- closed under $\cup$ and $\cap$, but **not** closed under $(\cdot)^c$:
  - monotone (exponential) Diophantine arithmetic
  - monotone AC-TA definable arithmetic  (cf. open question 2)
X. MTA and further extensions...
**Monotone A-tree automata**

\[ \text{ETA } (E, Q, Q_{\text{fin}}, \Delta) \]

- \( E \): \textbf{associative theory} \((F, E)\)
- \( F \): signature
- \( E \): finite set of associativity equations
  e.g. \((x + y) + z \approx x + (y + z)\)
- \( Q \): finite set of \textbf{state symbols} such that \( F \cap Q = \emptyset \)
- \( Q_{\text{fin}} \): finite set \( Q_{\text{fin}} (\subseteq Q) \) of \textbf{final states}
- \( \Delta \): finite set of transition rules with the following forms
  \[
  f(p_1, \ldots, p_n) \rightarrow q \quad \text{[regular rule]}
  \\
  p \rightarrow q \quad \text{[epsilon rule]}
  \\
  f(p_1, \ldots, p_n) \rightarrow f(q_1, \ldots, q_n) \quad \text{[monotone rule]}
  \]

**Question**

\[ C(\text{MA-TA}_E) \overset{?}{=} C(\text{RA-TA}_F) \]

**‘Monotone’ A-TA vs. ‘Regular’ A-TA**

Let \( E = (F, E) \)

\( E \): \((x + y) + z \approx x + (y + z)\)

\( F \): +
  (binary symbol)

\( a, b, c \cdots \) (constant symbols)

**Theorem**

For every A-TA \( \mathcal{A}_E \)

if it is monotone, the leaf language is \textbf{context-sensitive}

if it is regular, the leaf language is \textbf{context-free}

**Proof**

Due to the syntax, if \( \mathcal{A}_E \) is monotone, it has transition rules in the forms of \( p_1 + p_2 \rightarrow q_1 + q_2, \ p_1 + p_2 \rightarrow q \) and \( a \rightarrow q \ (a \in F) \). Looking at the rules from right to left, they correspond to rules of a context-sensitive grammar (Kuroda normal form, page 16 of seminar talk 2). Likewise, if \( \mathcal{A}_E \) is regular, the rules of \( \mathcal{A}_E \) correspond to rules of a context-free grammar in Chomsky normal form.
**A vs. AC**

Given a **monotone** AC-TA $A_{\mathcal{E}_1}$, one can construct a **monotone** A-TA $B_{\mathcal{E}_2}$ such that

$$\mathcal{L}(A_{\mathcal{E}_1}) = \mathcal{L}(B_{\mathcal{E}_2})$$

**Proof**

Let $A = (\mathcal{E}_1, Q, Q_{\text{fin}}, \Delta_1)$ and $\mathcal{E}_1 = (F, E_1)$. Then define $B = (\mathcal{E}_2, Q, Q_{\text{fin}}, \Delta_2)$ and $E_2 = (E, E_2)$ where

- $E_2 : E_1 - \{ f(x, y) \approx f(y, x) \mid f \in F \}$
- $\Delta_2 : \Delta_1 \cup \{ f(p, q) \rightarrow f(q, p) \mid p, q \in Q \}$

It is straightforward to show that $\mathcal{L}(B_{\mathcal{E}_2}) \subseteq \mathcal{L}(A_{\mathcal{E}_1})$. So it suffices to show the reverse inclusion $\mathcal{L}(A_{\mathcal{E}_1}) \subseteq \mathcal{L}(B_{\mathcal{E}_2})$. The property is obtained by the following simulation relations:

1. $=_{\mathcal{E}_1} \subseteq \rightarrow_{\Delta_2/\mathcal{E}_2}^*$
2. $\rightarrow_{\Delta_1} \subseteq \rightarrow_{\Delta_2}$

(Complete the proof in Exercise) □

**Closure properties (∪, ∩)**

The class of monotone A-TA is closed under union and intersection

**Proof**

It is obvious that the class is closed under union. A similar proof is found in page 9 of seminar talk 3. For the proof of the closure under intersection, one can apply the tree automata construction described in page 20 of seminar talk 9: Let $A_1 = (\mathcal{E}, P, P_{\text{fin}}, \Delta_1)$ and $A_2 = (\mathcal{E}, Q, Q_{\text{fin}}, \Delta_2)$ with $\mathcal{E} = (F, E)$. Suppose $F_\lambda$ is a set of all A symbols in $F$. Then define $\Delta_\lambda = \bigcup_{f \in F_\lambda} \Delta_f$ (instead of $\Delta_{AC}$).

$$\begin{align*}
\Delta_f : & \quad f(\langle p_1, q_1 \rangle, \langle p_2, q_2 \rangle) \rightarrow f(\langle p_1, q_1 \rangle, \langle o, q_2 \rangle) \quad \text{if } \exists f(p_1, p_2) \rightarrow p \in \Delta_1, q_1, q_2 \in Q \\
& \quad f(\langle p_1, q_1 \rangle, \langle p_2, q_2 \rangle) \rightarrow f(\langle p_3, q_1 \rangle, \langle p_4, q_2 \rangle) \quad \text{if } \exists f(p_1, p_2) \rightarrow f(p_3, p_4) \in \Delta_1, q_1, q_2 \in Q \\
& \quad f(\langle o, q_1 \rangle, \langle p_2, q_2 \rangle) \rightarrow (p_2, q) \quad \text{if } \exists f(q_1, q_2) \rightarrow q \in \Delta_2, p_2 \in P \\
& \quad f(\langle o, q_1 \rangle, \langle p_2, q_2 \rangle) \rightarrow f(\langle o, q_3 \rangle, \langle p_2, q_4 \rangle) \quad \text{if } \exists f(q_1, q_2) \rightarrow f(q_3, q_4) \in \Delta_2, p_2 \in P \\
& \quad \langle p_1, q_1 \rangle \rightarrow (p_2, q_1) \quad \text{if } \exists p_1 \rightarrow p_2 \in \Delta_1, q_1 \in Q \\
& \quad \langle p_1, q_1 \rangle \rightarrow (p_1, q_2) \quad \text{if } \exists q_1 \rightarrow q_2 \in \Delta_2, p_1 \in P \\
& \quad f(\langle p_1, q_1 \rangle, \langle o, q_2 \rangle) \rightarrow f(\langle o, q_1 \rangle, \langle p_1, q_2 \rangle) \quad \text{if } \exists p_1 \in P, q_1, q_2 \in Q \\
& \quad f(\langle o, q_1 \rangle, \langle p_1, q_2 \rangle) \rightarrow f(\langle p_1, q_1 \rangle, \langle o, q_2 \rangle) \quad \text{if } \exists p_1 \in P, q_1, q_2 \in Q.
\end{align*}$$

□
Complementation

The class of monotone A-TA is closed under complement

Proof

We show the proof for the case that \( F = \{ f, g \} \cup F_0 \), where \( f \) is A symbol and \( g \) is binary symbol. The proof can be generalized to a signature containing arbitrary many \( n \)-ary symbols \((n > 0)\) and A symbols: Let \( E = (F, E) \) with \( E = \{ f(f(x, y), z) \approx f(x, f(y, z)) \} \). Given \( A = (E, Q, Q_{\text{fin}}, \Delta) \), define \( B = (E', Q', Q'_{\text{fin}}, \Delta') \) as follows.

\[ Q' = 2^Q \]
\[ Q'_{\text{fin}} = \{ P \subseteq Q | P \cap Q_{\text{fin}} = \emptyset \} \]
\[ \Delta' = \{ a \rightarrow \{ p | a \rightarrow p \in \Delta \} | a \in F_0 \} \cup \Delta_1 \cup \Delta_2 \cup \Delta_3 \]

where

\[ \Delta_1 : g(P_1, P_2) \rightarrow P \]
if \( P_1, P_2 \subseteq Q \) and
\[ P = \{ p | g(p_1, p_2) \rightarrow p \in \Delta \text{ and } p_i \in P_i (1 \leq i \leq 2) \} \]

(Proof cont’d)


Proof (cont’d)

\[ \Delta_2 : f(\alpha, \beta) \rightarrow \gamma \text{ if } \gamma \rightarrow \alpha \beta \in G_P \]
\[ f(\alpha, \beta) \rightarrow f(\gamma, \delta) \text{ if } \gamma \delta \rightarrow \alpha \beta \in G_P \]
for all \((\emptyset \neq) P \subseteq Q\) where
\[ G_P : \text{ CSG such that} \]
\[ \mathcal{L}(G_P) = \bigcap_{p \in P} \mathcal{L}(\Phi(G_q)) - \bigcup_{p \in (Q - P)} \mathcal{L}(\Phi(G_q)) \]
\[ G_q : \text{ CSG whose start symbol is } q \text{ and production rules are} \]
\[ p_3 \rightarrow p_1 p_2 \text{ if } f(p_1, p_2) \rightarrow p_3 \in \Delta \]
\[ p_3, p_4 \rightarrow p_1 p_2 \text{ if } f(p_1, p_2) \rightarrow f(p_3, p_4) \in \Delta \]
\[ \Phi : \text{ mapping onto CSG that adds production rules} \]
\[ p \rightarrow S \text{ if } p \in S, S \subseteq Q \]

\[ \Delta_3 : f(\alpha, \beta) \rightarrow \gamma \text{ if } \gamma \rightarrow \alpha \beta \in G_0 \]
\[ f(\alpha, \beta) \rightarrow f(\gamma, \delta) \text{ if } \gamma \delta \rightarrow \alpha \beta \in G_0 \]

where
\[ G_0 : \text{ CSG such that} \]
\[ \mathcal{L}(G_0) = \{ w \in (2^Q)^* | |w| \geq 2 \} - \bigcup_{p \in Q} \mathcal{L}(\Phi(G_q)) \]

Using the above automaton, one can prove that \( \mathcal{L}(B_E) = (\mathcal{L}(A_E))^C \).
The membership problem of CSG:

instance is grammar $G = (\Sigma, T, N, q_0, \Delta)$, word $w$

answer is “yes” if $w \in L(G)$ ; “no” otherwise

This is PSPACE problem

**Proof**

This problem is $\leq_p^m$-reducible to the membership problem of LBA (linear bounded automata). The LBA membership problem is in NSPACE($n$) (Appendix in seminar talk 2). From Savitch’s theorem (page 17 in seminar talk 6), NPSPACE=PSPACE, and thus the CSG membership problem is in PSPACE.

**Note**

The membership problem of LBA is PSPACE-complete (Show that every PSPACE problem is $\leq_p^m$-reducible to this problem

Moreover, the (reverse) transformation from the LBA membership problem to the CSG membership problem is also $\leq_p^m$-reducible. Therefore, the membership problem of CSG is PSPACE-complete.

---

The membership problem of monotone A-TA is PSPACE-complete

**Proof**

PSPACE-hardness is an immediate consequence of the previous observation about the membership problem of CSG, because CSG can be simulated by monotone A-TA. On the other hand, since polynomially bounded recursion of PSPACE problem ($p^{PSPACE}$) is in PSPACE, so is the membership problem of monotone A-TA.

**Note 1**

The membership problem of regular A-TA is in P.

**Note 2**

The universality and inclusion problems of monotone A-TA are also undecidable.

($\because$ The emptiness problem of CSG is so.)

$\{w \in L(A_{\geq}) \land (L(A_{\geq})^c = \emptyset)\} \subseteq \emptyset$
The membership problem of monotone AC-TA is PSPACE-complete.

**Proof**

Since monotone AC-TA can be simulated by monotone A-TA (page 5) and the transformation can be done in polynomial time relative to the size of the input, the problem is in PSPACE. For PSPACE-hardness, one can use the reduction from the “quantified Boolean formula” problem (QBF):

instance is a first-order logical formula \( \psi \) in the syntax \( S \)

\[
S ::= x \mid \neg S \mid S \land S \mid \exists x(S)
\]

answer is “yes” if \( \psi \) is true; “no” otherwise

This problem is PSPACE-complete. So it suffices to show that for an arbitrary formula \( \psi \) in \( S \), one can construct a monotone AC-TA \( A_\varepsilon \) and a tree \( t \) such that \( \psi \) is true iff \( A_\varepsilon \) accepts \( t \). An example of such construction can be found in, e.g., our paper [Ohsaki & Talbot & Tison & Roos].


---

**Summary of decidability and closure properties**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( t \in L(A_\varepsilon) )?</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>( L(A_\varepsilon) = \emptyset ) ?</td>
<td>−</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>( L(A_\varepsilon) = T_F )?</td>
<td>−</td>
<td>?</td>
<td>−</td>
<td>✓</td>
</tr>
<tr>
<td>( L(A_\varepsilon) \subseteq L(B_\varepsilon) )?</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>✓</td>
</tr>
</tbody>
</table>

(✓: positive, −: negative, ?: unknown)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>closed under ( \cup )</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>closed under ( \cap )</td>
<td>✓</td>
<td>✓</td>
<td>−</td>
<td>✓</td>
</tr>
<tr>
<td>closed under ( ()^c )</td>
<td>✓</td>
<td>−</td>
<td>−</td>
<td>✓</td>
</tr>
</tbody>
</table>
Propositional tree automata (PTA)

$$\text{PTA } A = (\mathcal{E}, Q, \psi, \Delta)$$

$$\psi : \text{ propositional formula over } Q$$

$$t : \text{ accepted by } A^e \text{ if } \{q \in Q \mid t \rightarrow A^e q\} \models \psi$$

where

$$P \models q \text{ if } q \in P$$

$$P \models \psi_1 \land \psi_2 \text{ if } P \models \psi_1 \land P \models \psi_2$$

$$P \models \neg \psi \text{ if } P \not\models \psi$$

Note

The class of PTA is closed under Boolean operations. (Exercise)

The membership problem of PTA is decidable if \(\{q \in Q \mid t \rightarrow A^e q\}\) is computable.
(In particular, if \(\mathcal{E}\) is AC theory, the membership problem is in \(\Delta^P_2\).)


Example

Consider (regular) PTA \(A\) with

$$\Delta : a \rightarrow q_a \quad b \rightarrow q_b \quad f(q_a, r) \rightarrow q_a \quad f(r, r) \rightarrow r \quad f(r, q_b) \rightarrow q_b$$

$$a \rightarrow r \quad b \rightarrow r \quad c \rightarrow r$$

$$\psi : q_a \land \neg q_b$$

\(A\) accepts trees whose leftmost leaf is ‘a’ and rightmost is not ‘b’

### Note

<table>
<thead>
<tr>
<th></th>
<th>M A-PTA</th>
<th>M AC-PTA</th>
<th>R A-PTA</th>
<th>R AC-PTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(t \in \mathcal{L}(A^e))?</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(\mathcal{L}(A^e) = \emptyset)?</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>(\mathcal{L}(A^e) = T_e)?</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>(\mathcal{L}(A^e) \subseteq \mathcal{L}(B^e))?</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>✓</td>
</tr>
</tbody>
</table>

(✓ : positive, – : negative)
Tree automata with normalization (TAN)

TAN \( A = (R, Q, Q_{\text{fin}}, \Delta) \)

- \( R \): equational rewrite system \((\mathcal{E}, R)\)
- \( t \): accepted by \( A_R \) if \( \exists u: t \rightarrow^{1}_R u \ \& \ u \rightarrow^{*}_{A_E} q \ (q \in Q_{\text{fin}}) \)

Note

The class of TAN is closed under Boolean operations if \( R \) is confluent & terminating.

The membership problem of TAN is decidable if \( R \) is terminating & the membership problem of \( A_E \) is decidable. (relative decidability)

Question

Let \( R_I = \{ x + x \rightarrow x \} \) and \( E_{AC} = \{ (x + y) + z \approx x + (y + z) \ x + y \approx y + x \} \)

Is it decidable for TAN \( A_R \) with \( R_I \) and \( E_{AC} \), whether \( \mathcal{L}(A_R) = \emptyset \)?


Idempotency \( X + X \approx X \)

Let \( E_{ACI} = E_{AC} \cup \{ s(x) + s(x) \approx s(x) \} \)

Consider tree language \( L : s^i(0) + \cdots + s^j(0) \ (i \in \mathbb{N}) \)

- \( L \) is accepted by ETA with \( \mathcal{E} = (F, E_{ACI}) \)
- \((L)^C\) is not accepted by ETA with \( \mathcal{E} = (F, E_{ACI}) \)

(\( \vdash \): Suppose \( A_{\mathcal{E}} \) accepts \( (L)^C \). From Pumping Lemma, there exist \( m, n \ (m \neq n) \) such that \( s^m(0) \rightarrow^{*}_{A_{\mathcal{E}}} q \) and \( s^n(0) \rightarrow^{*}_{A_{\mathcal{E}}} q \ (q \in Q) \). Since \( A_{\mathcal{E}} \) accepts \( s^m(0) + s^n(0) \), \( A_{\mathcal{E}} \) accepts also \( s^m(0) + s^n(0) \), leading to the contradiction.)

Let \( R_{ACI} = (F, E_{AC}, \{ s(x) + s(x) \rightarrow s(x) \}) \)

- \((L)^C\) is accepted by TAN with \( R_{ACI} \):
  \[
  0 \rightarrow q \quad s(q) \rightarrow q \quad q + q \rightarrow q_f \quad s(q_f) \rightarrow q \quad q + q_f \rightarrow q_f \quad (q_f: \text{final state})
  \]

If eliminate the last two rules, the TAN accepts:

\[
  s^{m_1}(0) + \cdots + s^{m_k}(0) \quad (m_i \neq m_j \text{ for some } i, j)
\]
Exercise

1. Show that $\varepsilon_1 \subseteq \Delta_2^\ast / \varepsilon_2$ in page 5.

2. Show that every PSPACE problem is $\leq_P^m$-reducible to the membership problem of LBA (page 9). (Hint: For a problem with input of the size $n$ and with worktape of the length bounded by a polynomial $P(n)$, consider how to simulate the computation of the problem by LBA.)

3. Show that the class of monotone A-TA subsumes the class of the Boolean closure $Cl_{\text{Bool}}(M \text{ AC-TA})$ of monotone AC-TA.

4. Show that the class of PTA is closed under Boolean operations (both in regular and monotone cases).

5. Show that (1) the emptiness problem of (regular) AC-PTA is decidable, and (2) the emptiness problem of monotone AC-PTA is undecidable.

6. Show that the class of TAN with ETRS $\mathcal{R}$ is closed under Boolean operations if $\mathcal{R}$ is terminating and confluent.

7. Show that $\mathcal{R} = (F, E_{AC}, R_I)$ in page 15 is terminating and confluent.

Appendix: Overview of project ETA ($\eta$)