From Inclusion to Universality (the Cool Way)

Goal: · Previous problem was $\Sigma^\omega = L(A)$ $(\subseteq)$

· Solve more general problem

$$L(A) \subseteq L(B) \text{ for arbitrary NBAs } A, B$$

(above fixes $\Sigma^\omega = L(\rightarrow\circledcirc \Sigma)$)

Approach: Reduce inclusion to universality

Claim:

Let A, B two NBAs over alphabet Σ.

There is an NBA C over an alphabet Δ so that

$$L(A) \subseteq L(B) \quad \text{iff} \quad L(C) = \Delta^\omega$$

Moreover, the size of C is polynomial in the sizes of A and B.

Approach:

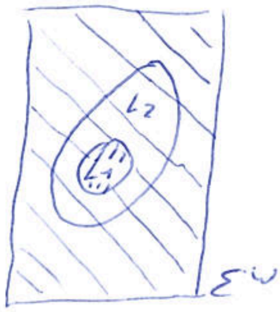
Consider L_1 and L_2 over alphabet Σ. Then

$$L_1 \subseteq L_2$$

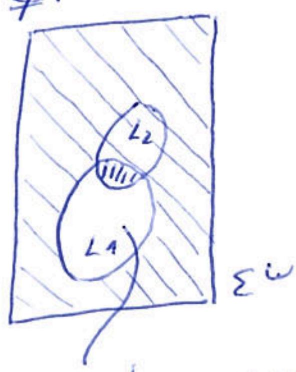
iff $L_1 \cap L_2 = L_1$

iff $(L_1 \cap L_2) \cup \overline{L_1} = \Sigma^\omega$

$\subseteq$:



$\not\subseteq$:



This part is missing if inclusion fails.

Problem:

For a word, a single run of A does not reveal whether or not the word is accepted (there may be another run that is accepting)

Idea:

But for every run, one can of course say whether or not it is accepting

↳ Base the technical development on <u>runs</u> of A (instead of words)

↳ Let automaton C work on $\underbrace{\text{letters + states}}_{\text{new alphabet}}$ of A

Construction:

- Let $\mathcal{A} = (\Sigma, Q_{\mathcal{A}}, q_0^{\mathcal{A}}, \to_{\mathcal{A}}, Q_F^{\mathcal{A}})$ and $\mathcal{B} = (\Sigma, Q_{\mathcal{B}}, q_0^{\mathcal{B}}, \to_{\mathcal{B}}, Q_F^{\mathcal{B}})$
- To make $\mathcal{C}$ work on the runs of $\mathcal{A}$, extend the alphabet:

$$\Delta := \Sigma \times Q_{\mathcal{A}}$$

- We have

$$L(\mathcal{A}) \subseteq L(\mathcal{B})$$

iff for every accepting run of $\mathcal{A}$

$$r_{\mathcal{A}} = q_0^{\mathcal{A}} \xrightarrow{a_0} q_1^{\mathcal{A}} \xrightarrow{a_1} q_2^{\mathcal{A}} \xrightarrow{a_2} \dots$$

there is an accepting run of $\mathcal{B}$:

$$r_{\mathcal{B}} = q_0^{\mathcal{B}} \xrightarrow{a_0} q_1^{\mathcal{B}} \xrightarrow{a_1} q_2^{\mathcal{B}} \xrightarrow{a_2} \dots$$

- If we rewrite the implication, this can be equivalently expressed as

for every sequence

$$q_0^{\mathcal{A}} \xrightarrow[q_1^{\mathcal{A}}]{a_0} q_1^{\mathcal{A}} \xrightarrow[q_2^{\mathcal{A}}]{a_1} q_2^{\mathcal{A}} \xrightarrow[q_3^{\mathcal{A}}]{a_2} \dots \quad \text{in } \Delta^\omega$$

we have

(1) the sequence is no accepting run of $\mathcal{A}$, because

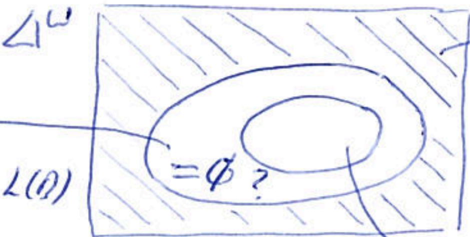
(1a) it is no run of $\mathcal{A}$

(1b) it is not accepting.

or

(2) $a_0 a_1 a_2 \dots \in L(\mathcal{B})$.

Illustration:



accepting runs of $\mathcal{A}$ on words outside $L(\mathcal{B})$

Sequences that are (1a) no runs of $\mathcal{A}$ (1b) or not accepting

(2) accepting runs of $\mathcal{A}$ on words that are also in $L(\mathcal{B})$.

Inclusion $L(\mathcal{A}) \subseteq L(\mathcal{B})$ holds iff $= \emptyset$ is true, i.e., (1a), (1b), (2) cover Δ^ω.

Construct C so that it accepts the Δ^ω words that satisfy (1a), (1b), or (2).

This means

$$L(A) \subseteq L(B) \quad \text{iff} \quad L(C) = (\Sigma \times Q_A)^\omega = \Delta^\omega.$$

Automaton C:

(2)

$A \times B$ on $\Sigma \times Q_A$
with

- $(q_A, q_B) \xrightarrow[q_A']{a}_C (q_A', q_B')$

if $q_A \xrightarrow{a}_A q_A'$ and $q_B \xrightarrow{a} q_B'$

- (q_A, q_B) is final if q_B is final in B

A fails to do a transition

(1a)

guess last accepting state of A

(1b)

- A with final states removed
- Remaining states turned into final states

Behavior:

C guesses run of A and B on word w

↳ Check that run respects transitions of A

⇒ No: accept by (1a)

⇒ Yes: Guess whether A accepts

⇒ A accepts: B has to accept (2)

⇒ A does not accept

↳ Guess last accepting state of A

↳ Accept from now on (1b).

6. Linear-time temporal logic

- Specification language for model checking:

  In $A \models \varphi$, formula φ is typically in LTL.

- Used in industry (PSL = property specification language, variant of LTL, like statemachines in UML are derived from finite automata).

- Found by Amir Pnueli '77, Turing award 1996

Idea of LTL:

↳ Subset of MSO useful for specification

↳ No quantifiers, more complex and intuitive operators

↳ View word as a sequence of (sets of) actions over time

↳ Interpret formula at a single moment / point in the word

$$w = \underline{\quad \alpha \quad \overset{a}{|} \quad \beta \quad}$$

$\Rightarrow$ a is now

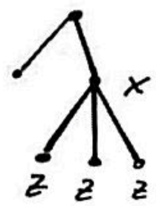
$\Rightarrow$ β is future

↳ Operators make claims about the future.

<u>Remark:</u>

- LTL is a <u>linear-time</u> temporal logic that talks about words
- CTL is a <u>branching-time</u> temporal logic that talks about computation trees:

$$EO(x \wedge AO z)$$



- CTL* unifies (and generalises) LTL and CTL.

<u>Goal:</u>

↳ Translate LTL into NBA for model checking

↳ LTL can be seen as a subset of MSO

$\Rightarrow$ We know that it can be done

↳ But it is strictly less expressive than MSO

$\Rightarrow$ We obtain a faster and easier algorithm.

<u>6.1 Syntax and semantics of LTL</u>

<u>Recall:</u> For translation of WMSO formulas $\varphi(X_1, \dots, X_n)$ we had NFAs over $\{0,1\}^n$, vectors of Booleans.

<u>In LTL:</u>
- Finite set of <u>propositions</u> $p, q, \dots \in P$
  - ↳ Imitate second-order variables X_i
  - ↳ Finite in every system
- Define <u>alphabet</u> $\Sigma = \mathbb{P}(P)$
  - ↳ letters again yield vectors:

$$a \in \Sigma \text{ means } a \subseteq P \text{ with } a = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \begin{array}{l} p_1 \in a \\ p_2 \notin a \\ \\ p_n \in a \end{array}$$

↳ but we can use set notation, $p \in a$.

- Practically, system · does more than one action at a time
  · has components that are in one state each

Definition (Syntax of LTL):

Let $\mathcal{P} = \{p, q, \ldots\}$ a finite set of propositions.
Formulas in LTL over $\Sigma = \mathbb{P}(\mathcal{P})$ are defined by

$$\varphi ::= p \mid \varphi \vee \psi \mid \neg\varphi \mid \underbrace{\bigcirc\varphi}_{\text{"next"}} \mid \underbrace{\varphi \mathcal{U} \psi}_{\text{"until"}}$$

with $p \in \mathcal{P}$. Use following abbreviations (besides standard abbreviations for Boolean operators):

$$\underbrace{\Diamond\varphi}_{\text{"eventually"}} = true\ \mathcal{U}\ \varphi \qquad \underbrace{\Box\varphi}_{\text{"globally"}} = \neg\Diamond\neg\varphi$$

$$\underbrace{\varphi R \psi}_{\text{"release"}} = \neg(\neg\varphi\ \mathcal{U} \neg\psi)$$

Intuition:

- p = proposition p holds at the current position
- $\bigcirc\varphi$ = the next position satisfies φ
- $\varphi\mathcal{U}\psi$ = ↳ φ holds in all positions until ψ holds
  ↳ And ψ definitely holds some time later (or already now)
- $\Diamond\varphi$ = there is some future moment in which φ holds
- $\Box\varphi$ = from now on, φ holds in all moments in the future
- $\varphi R \psi$ = dual of until ↳ ψ holds as long as it is not released by φ
  ↳ ψ may hold forever or
  ↳ there is a moment with ψ and φ.

Definition (Satisfaction relation):

Let $w = a_0 a_1 \ldots \in \Sigma^\omega = \mathbb{P}(\mathcal{P})^\omega$. The satisfaction relation $\models$ is defined inductively as follows (for all $i \in \mathbb{N}$):

$w, i \models p$ if $p \in a_i$

$w, i \models \varphi \vee \psi$ if $w, i \models \varphi$ or $w, i \models \psi$

$w, i \models \neg\varphi$ if not $w, i \models \varphi$

$w, i \models \bigcirc\varphi$ if $w, i+1 \models \varphi$

$w, i \models \varphi\, \mathcal{U}\, \psi$ if there is $k \geq i$ so that

- for all $i \leq j < k$ we have $w, j \models \varphi$
- $w, k \models \psi$.

An LTL-formula φ <u>defines a language</u> $\mathcal{L}(\varphi) \subseteq \Sigma^\omega$ by interpreting it in the first position of a word:

$$w \models \varphi \quad \text{if} \quad w, 0 \models \varphi$$

and

$$\mathcal{L}(\varphi) := \{ w \in \Sigma^\omega \mid w \models \varphi \}.$$

<u>Examples:</u>

- Infinitely often φ: $\Box\Diamond\varphi$
- Finitely often φ: $\Diamond\Box\neg\varphi$
- If there are infinitely many positions with p, then there are infinitely many positions with q:
  $$\Box\Diamond q \vee \Diamond\Box\neg p$$
- Every request is followed by an acknowledge:
  $$\Box(req \rightarrow \Diamond ack).$$

<u>Note:</u>

- Assume your system creates entities (object-oriented programs with new)
- Then LTL is not sufficient to express correctness.

$$\forall \text{components } c: \Box(req@c \Rightarrow ack@c).$$

Defining such a logic + basic decision procedures could be a Master's thesis.

Recapitulation:

Goal: $\mathcal{A} \models \varphi$ with φ in LTL

Definition of LTL:

- Finite set of propositions $(p, q, \ldots \in)$ $\mathcal{P}$
- Letters are sets of propositions, $\Sigma = \mathcal{P}(\mathcal{P})$ $(\ni a)$
- Formulas evaluated in positions of words w in Σ^ω:

  $w, i \models \varphi$.

Syntax of LTL:

$$\varphi ::= p \mid \varphi \vee \psi \mid \neg\varphi \mid \underbrace{\bigcirc\varphi}_{\text{"next } \varphi\text{"}} \mid \underbrace{\varphi \mathcal{U} \psi}_{\text{"}\varphi \text{ until } \psi\text{"}}$$

Abbreviations:

$$\underbrace{\Diamond\varphi}_{\text{"eventually } \varphi\text{"}} := true\, \mathcal{U} \varphi \qquad \underbrace{\Box\varphi}_{\text{"globally / always } \varphi\text{"}} := \neg\Diamond\neg\varphi$$

$$\underbrace{\varphi \mathcal{R} \psi}_{\text{"}\varphi \text{ releases } \psi\text{"}} := \neg(\neg\varphi \mathcal{U} \neg\psi)$$

Size of a formula:

$|p| := 1 \qquad |\neg\varphi| := 1 + |\varphi| =: |\bigcirc\varphi|$

$|\varphi \overset{\vee}{\underset{\mathcal{U}, \mathcal{R}}{\wedge}} \psi| := |\varphi| + 1 + |\psi|$.

Logical equivalence $\equiv$:

$\varphi \equiv \psi$ iff for all $w \in \Sigma^\omega$ and all $i \in \mathbb{N}$

$w, i \models \varphi$ iff $w, i \models \psi$.

Some language-theoretic considerations:

Every letter $a \in \Sigma$ can be described precisely by characteristic formula:

$$\chi_a := \bigwedge_{p \in a} p \wedge \bigwedge_{p \notin a} \neg p.$$

With this, capture languages over Σ by LTL formulas.

Language $(ab)^\omega$ defined by

$$x_a \wedge \Box\left((x_a \to \bigcirc x_b) \wedge (x_b \to \bigcirc x_a)\right).$$

Language $\underbrace{(a.(a \cup b))^\omega}_{\text{"even positions have an a"}}$ not LTL-definable.

Why?

↳ LTL-definable languages definable in FO on infinite words

↳ Recall:

↳ Words of even length not definable in FO on finite words

↳ Similar argument here.

Positive normal form and properties of until

Definition (Positive normal form):

Let $\mathcal{P}$ a finite set of propositions.

The LTL formula over $\Sigma = \mathcal{P}(\mathcal{P})$ is in positive normal form if it is constructed from

$p, \neg p$ with $p \in \mathcal{P}$ and $\vee, \wedge, \bigcirc, \mathcal{U}, \mathcal{R}$.

Lemma:

Every LTL formula φ over Σ is logically equivalent to a formula ψ in positive normal form with $|\psi| \leq 2|\varphi|$.

Proof:

Use following equivalences:

- $\neg \bigcirc \varphi \equiv \bigcirc \neg \varphi$
- $\neg(\varphi \mathcal{U} \psi) \equiv \neg(\neg(\neg\varphi) \mathcal{U} \neg(\neg\psi)) \equiv \neg\varphi \mathcal{R} \neg\psi$
- $\neg(\varphi \mathcal{R} \psi) \equiv \neg\varphi \mathcal{U} \neg\psi.$

$\square$

For translation of LTL into Büchi automata,
use "unrolling" of $\mathcal{U}$ (also called inductive property)

Lemma:
For all $\varphi, \psi \in LTL$ we have $\varphi \mathcal{U} \psi \equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \mathcal{U} \psi))$.

Proof: Homework.

Logical equivalence $\equiv$ in LTL is in fact a congruence:
If $\varphi \equiv \psi$ and φ is part of a larger formula $\Theta(\varphi)$,
then $\Theta(\varphi) \equiv \Theta(\psi)$.

As a consequence:

$$\begin{aligned} \varphi \mathcal{U} \psi &\equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \mathcal{U} \psi)) \\ &\equiv \psi \vee (\varphi \wedge \bigcirc(\psi \vee (\varphi \wedge \bigcirc(\varphi \mathcal{U} \psi)))) \\ &\equiv \dots \end{aligned}$$

Gives a means to check $\varphi \mathcal{U} \psi$ at position i:
- either ψ holds
- or φ holds and $\varphi \mathcal{U} \psi$ holds in next position $i+1$.

Have to ensure ψ eventually holds (unrolling happens only finitely many times).
↳ Find states forbid infinite unrollings.
⇒ Following procedure exploits unrolling.

6.2 From LTL to NBA

Goal: Translate LTL into NBA
↳ without using intermediary FO representation
↳ and then Büchi's result.
Why is LTL easier?
↳ Like automaton only looks into future
↳ Do not follow inductive structure of formulas
⇒ saves determinisation / complementation

at each negation

$\Rightarrow$ and thus exponential blow-ups.

$\hookrightarrow$ Instead, keep track of satisfaction of all subformulas while reading input.

<u>Definition (Generalized NBA):</u>

A <u>generalised nondeterministic Büchi automaton</u> GNBA is a tuple

$$A = (Q, Q_I, \rightarrow, (Q_F^i)_{1 \le i \le k})$$

with

- <u>set of initial states</u> $Q_I \subseteq Q$ (instead of $q_0 \in Q$)
- <u>family of final states</u> $(Q_F^i)_{1 \le i \le k}$

A <u>run</u> is still

$$r = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} \dots \quad \text{with } q_0 \in Q_I.$$

A run is <u>accepting</u> if

$$Inf(r) \cap Q_F^i \neq \emptyset \quad \text{f.a. } 1 \le i \le k$$

"Every set of final states is visited infinitely often"

"Generalization" does not increase expressiveness of the automaton model.

<u>Lemma:</u>

For every GNBA $A = (Q, Q_I, \rightarrow, (Q_F^i)_{1 \le i \le k})$ there is an NBA $A' = (Q', q_0, \rightarrow', Q_F')$ with $L(A) = L(A')$ and $|Q'| = |Q| k + 1$.

<u>Idea:</u>

Use counters from intersection construction:

$$L(A) = \bigcap_{1 \le i \le k} L(A_i) \quad \text{with } A_i = (Q, Q_I, \rightarrow, Q_F^i)$$

Construction (directly):

↳ Several initial states to one ⇝ pic new state

↳ Several sets of final states to one:

⇒ use counters in new states: $Q' = Q \times \{1, \dots, k\}$

⇒ (q, i) means: next final state is expected from Q_F^i.

⇒ new final states: $Q_F^1 \times \{1\}$

(this choice is arbitrary, could be any $1 \le i \le k$). □

Idea of the translation:

↳ Subformulas of $\theta \in LTL$ as states in the automaton

↳ Intuition: formulas that currently hold.

Definition (Fischer-Ladner closure):

Let θ an LTL formula in positive normal form.

Its Fischer-Ladner closure $FL(\theta)$ is the smallest set of LTL formulas in positive normal form so that

- $\theta \in FL(\theta)$ and
- If $\varphi \overset{\vee}{\wedge} \psi \in FL(\theta)$ then $\{\varphi, \psi\} \subseteq FL(\theta)$.
- If $\varphi \mathcal{U} \psi \in FL(\theta)$ then $\psi \vee (\varphi \wedge O(\varphi \mathcal{U} \psi)) \in FL(\theta)$.
- If $\varphi \mathcal{R} \psi \in FL(\theta)$ then $\psi \wedge (\varphi \vee O(\varphi \mathcal{R} \psi)) \in FL(\theta)$.
- If $O\varphi \in FL(\theta)$ then $\varphi \in FL(\theta)$.

Example:

Let $\theta = p \,\mathcal{U} \neg p$. Then

$$FL(\theta) = \{p \,\mathcal{U} \neg p,\ \neg p \vee (p \wedge O(p \,\mathcal{U} \neg p)),\ \neg p,\ p \wedge O(p \,\mathcal{U} \neg p),\ p,\ O(p \,\mathcal{U} \neg p)\}$$

- Definition of Fischer-Ladner closure purely syntactical
- Following definition yields subsets A closed under "satisfaction of subformulas" ("what else has to hold")

$\hookrightarrow$ If $\varphi \vee \psi \in M$ then $\varphi \in M$ or $\psi \in M$.

Single out those sets that do not contain contradictions p and $\neg p$.

<u>Definition (Hintikka set)</u>:

Let θ an LTL formula in positive normal form.

A <u>Hintikka set</u> for θ is a subset $M \subseteq FL(\theta)$

so that the following closure properties hold:

- $\varphi \vee \psi \in M$ implies $\varphi \in M$ or $\psi \in M$
- $\varphi \wedge \psi \in M$ implies $\varphi \in M$ and $\psi \in M$
- $\varphi \mathcal{U} \psi \in M$ implies $\psi \in M$ or ($\varphi \in M$ and $O(\varphi \mathcal{U} \psi) \in M$)
- $\varphi \mathcal{R} \psi \in M$ implies $\psi \in M$ and ($\varphi \in M$ or $O(\varphi \mathcal{R} \psi) \in M$).

A Hintikka set $M \subseteq FL(\theta)$ is called <u>consistent</u>

if there is no $p \in \mathcal{P}$ with $\{p, \neg p\} \subseteq M$.

Denote by $\mathcal{H}(\theta)$ the set of <u>all consistent Hintikka sets</u>

for θ.

Define

$\mathcal{P}^+(M) := M \cap \mathcal{P}$ // set of propositions that occur <u>positively</u> in M.

$\mathcal{P}^-(M) := \{p \in \mathcal{P} \mid \neg p \in M\}$ // set of propositions that occur <u>negatively</u> in M.

<u>Example (continued)</u>:

Let $\theta = p \mathcal{U} \neg p$.

Then $\mathcal{H}(\theta) = \{\emptyset, \{p\}, \{\neg p\}, \{p, O(p \mathcal{U} \neg p)\},$

$\{p \mathcal{U} \neg p, p, O(p \mathcal{U} \neg p)\}, \dots\}$