2. Weak monadic second-order logic

Goal: Solve decidability problems in logic with help of automata

↳ validity / satisfiability of formulas.

Fix alphabet Σ.

2.1 Syntax and semantics

Need signature $Sig = (Fun, Pred)$

⇒ Have, purely relational signatures with $Fun = \emptyset$

⇒ Fix $Pred = \{ </2, suc/2, P_a/1 \text{ with } a \in \Sigma \}$

Definition (WMSO):

Consider two countably infinite sets

- $V_1 = \{x, y, z, \dots\}$ of first-order variables (small letters)
- $V_2 = \{X, Y, Z, \dots\}$ of second-order variables (capital letters)

Formulas in WMSO (over Sig, V_1, V_2) are defined by

$$\varphi ::= \underbrace{x<y \mid suc(x,y) \mid P_a(x)}_{\text{predicates from signature}} \mid X(x) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \exists x: \varphi \mid \exists X: \varphi$$

Sometimes make signature explicit. Write

WMSO$[<, suc]$ for all WMSO formulas,

WMSO$[<]$, WMSO$[suc]$ if we restrict ourselves to predicate $<$ or suc, respectively.

Similarly:

FO$[<, suc]$, FO$[<]$, FO$[suc]$ for first-order formulas over V_1 (without V_2 variables).

Intuitively:

- First-order variables ⇝ natural numbers $\mathbb{N}$

  $x < y$ ⇝ usual $<$ on $\mathbb{N}$

- Second-order variables ⇝ finite sets of natural numbers

  $X(x)$ ⇝ x is in X.   Weak MSO   monadic ⇝ polyadic = sets of tuples (relations)

Abbreviations:

$\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$ $\varphi \to \psi := \neg\varphi \vee \psi$

$\forall x: \varphi := \neg\exists x: \neg\varphi$ $\forall X: \varphi := \neg\exists X: \neg\varphi$

$x \leq y := \neg(y < x)$ $x = y := x \leq y \wedge y \leq x$.

$first(x) := \neg\exists y: y < x$ $last(x) := \neg\exists y: x < y$.

Example:

- $\exists X: (\exists x: first(x) \wedge X(x)) \wedge (\forall x: X(x) \to \exists y: x < y \wedge X(y))$

  There is a finite set of numbers
  - that contains 0 (and thus is not empty) and
  - for every element contains a larger one.

  Such a set does not exist (has to be infinite ↯).

  ⇒ Formula false (unsatisfiable)

- $\exists x, x < x$

  ⇒ Also false (unsatisfiable).

Bound and free variables:

- Variable x (and X) called bound if

  syntax tree contains occurrence of $\exists x$ ($\exists X$) above x (X)
- Otherwise free
- Write $\varphi(x_1, \ldots, x_m, X_1, \ldots, X_n)$ to indicate that free variables of φ are among $x_1, \ldots, x_m, X_1, \ldots, X_n$.
- Formula without free variables called closed, or sentence
- Assume bound and free variables disjoint

  $\psi = (x < z) \wedge (\forall x: x < y)$ (bad)

  Can always be achieved by α-conversion (of bound variables)

  $\psi' = (x < z) \wedge (\forall x': x' < y)$. (good)

Example:

- $\underbrace{\neg \exists y: y < x}_{first(x)}$ $\leadsto$ x free, y bound
- $\exists x: first(x) \wedge X(x)$ $\leadsto$ x bound, X free.

To define semantics, we need Sig-structures

$$S = (D_S, \underbrace{<_S, suc_S, (P_{a_S})_{a \in \Sigma}}_{\text{interpretation of predicate symbols}})$$

domain = elements to quantify over

$<_S, suc_S \subseteq D_S \times D_S$, $P_{a_S} \subseteq D_S$

Here: word structures

$w = a_0 \dots a_{n-1}$ as function from positions $\{0, \dots, n-1\}$ to letters Σ

Definition (Word structure):

Let $w = a_0 \dots a_{n-1}$. Its word structure is

$$S_w = (D_w, <_w, succ_w, (P_{a_w})_{a \in \Sigma})$$

with

$D_w := \{0, \dots, n-1\}$

$<_w := <_{\mathbb{N}} \cap (D_w \times D_w)$

$succ_w := \{(0,1), (1,2), \dots, (n-2, n-1)\}$

$P_{a_w} := \{\underbrace{h \in D_w \mid w(h) = a}_{\text{Positions in } w \text{ with } a.}\}$

Definition (Satisfaction relation $\models$ for WMSO):

Let $w \in \Sigma^*$ and φ a WMSO formula.

Let $I: V_1 \cup V_2 \to D_w \cup \mathbb{P}(D_w)$

then $S_w, \mathcal{I} \models P_a(x)$ if $\mathcal{I}(x) \in P_{aw}$
(also write $P_{aw}(\mathcal{I}(x))$)

$S_w, \mathcal{I} \models suc(x,y)$ if $(\mathcal{I}(x), \mathcal{I}(y)) \in suc_w$
(also $suc_w(\mathcal{I}(x), \mathcal{I}(y))$)

$S_w, \mathcal{I} \models x < y$ if $\mathcal{I}(x) <_w \mathcal{I}(y)$

$S_w, \mathcal{I} \models X(x)$ if $\mathcal{I}(x) \in \mathcal{I}(X)$

$S_w, \mathcal{I} \models \varphi_1 \vee \varphi_2$ if $S_w, \mathcal{I} \models \varphi_1$ or $S_w, \mathcal{I} \models \varphi_2$

$S_w, \mathcal{I} \models \neg\varphi$ if not $S_w, \mathcal{I} \models \varphi$

$S_w, \mathcal{I} \models \exists x: \varphi$ if there is $k \in D_w$ so that $S_w, \mathcal{I}[k/x] \models \varphi$

$S_w, \mathcal{I} \models \exists X: \varphi$ if there is $M \subseteq D_w$ so that $S_w, \mathcal{I}[M/X] \models \varphi$

Here, $\mathcal{I}[k/x](y) := \mathcal{I}(y)$ if $y \neq x$ and $\mathcal{I}[k/x](x) := k$.
Similar for X.

- Interested in closed-formulas (sentences)
  - ↳ Meaning does not depend on interpretation $\mathcal{I}$ of variables.
  - ↳ Needed interpretation for truth of subformulas.

- Say that closed-formula φ is satisfiable if $S_w \models \varphi$ for some $w \in \Sigma^*$
- Call S_w (or w) a model of φ.
- Formula without model is unsatisfiable.
- If $S_w \models \varphi$ for all $w \in \Sigma^*$, then φ is valid

Note:

φ valid iff $\neg\varphi$ unsatisfiable

Set of words that satisfy (are models of) φ form a language.

Definition (Language defined by a formula):
Let φ a closed WMSO-formula.
- The language defined by φ is

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

- A language $L \subseteq \Sigma^*$ is called <u>WMSO-definable</u> if there is a formula φ with $L = L(\varphi)$.
- Notions like <u>WMSO[suc]</u>, <u>FO[suc]-definable</u> are defined by restricting φ.

<u>Examples:</u>

(a) For $\psi = \forall x : \forall y : (P_a(x) \wedge P_b(y)) \to x < y$
we have
$$L(\psi) = a^* . b^*.$$
So $L(\psi)$ is FO[<]-definable.

(b) Language $\Sigma^* . a . b . \Sigma^*$ is WMSO[suc] (in fact FO[suc])-definable by
$$\exists x : \exists y : P_a(x) \wedge P_b(y) \wedge suc(x,y)$$

(c) $L = \{ w \in \{a,b\}^* \mid |w| \text{ is odd} \}$ is WMSO[suc]-definable by

$$\underset{\text{even positions}}{\exists X} : \quad \exists x : first(x) \wedge X(x) \wedge \underbrace{\forall x : \forall y : suc(x,y) \to (X(x) \leftrightarrow \neg X(y))}_{\text{every second position is in } X} \wedge \underbrace{\exists y : last(y) \wedge X(y)}$$

If last letter has an even position, word has odd length (as starts with 0).
Consider
$$aba.$$
We have
$$|aba| = 3$$
and
$$ab\underset{\hookrightarrow \text{ Position } 2.}{a}$$

(d) $L = \{w \in \{a,b,c\}^* \mid$ for every a that you see, there will follow as until $a b$ occurs and b definitely occurs. $\}$

is WMSO-definable.

$$\forall x: P_a(x) \to \exists y: x < y \wedge P_b(y) \wedge \forall z: x < z \wedge z < y \to P_a(z).$$

Distinguish between

$\overset{FO}{WMSO}[<]$, $\overset{FO}{WMSO}[<, suc]$, $\overset{FO}{WMSO}[suc]$ - definability.

<u>Lemma</u>

(a) L is $FO[<, suc]$-definable iff L is $FO[<]$-definable.

(b) L is $WMSO[<, suc]$-definable iff L is $WMSO[<]$-definable

(c) L is $WMSO[<, suc]$-definable iff L is $WMSO[suc]$-definable.

(d) Let $WMSO_0$ = WMSO without first-order variables.

New atomic formulas for $WMSO_0$:

$X \subseteq Y$, $Sing(X)$, $Suc(X, Y)$, $X \subseteq P_a$ $(a \in \Sigma)$

with the following meaning:

- X is subset of Y,
- X is a singleton set
- X and Y are singletons $X = \{x\}$, $Y = \{y\}$ so that $suc(x, y)$
- X is a subset of P_a

Then

L is $WMSO[<, suc]$-definable iff L is $WMSO_0$-definable

<u>Proof</u>:

(a) and (b): $S_w, I \models suc(x,y)$ iff $S_w, I \models x < y \wedge \neg\exists z. x < z \wedge z < y.$

(c) and (d): Homework.

For (d): Interpret first-order variables as singletons.

Further relations:

- WMSO vs. FO: later
- $FO[suc]$ strictly weaker than $FO[<]$ (not here).

2.2 Büchi's theorem

Relationship between WMSO-definability and regularity.

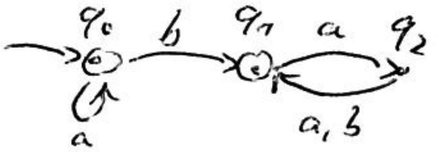
2.2.1 From automata to logic

Theorem (Büchi I):

For every regular language L, we can effectively construct a WMSO formula φ_L with $L = L(\varphi_L)$.

Illustration.

A



Consider $w = baaab \in L(A)$.

Goal: $S_w \models \varphi_{L(A)}$

Idea:

- Encode full runs, including information about states

$$q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_1 \in Q_F.$$

- Second-order variables for states
  - ↳ X_j for moments we are in q_j.

Problem:

Second-order variables only for letter positions

Solution:

↳ Encode moments indirectly

↳ Let X_j store positions of letters that stem from q_j

⟺ If $h \in X_j$, then we are in q_j before the letter at h is taken.

⟺ If $h \in X_j$, then we are in q_j after letters $0 \ldots h-1$.

In the example:

Run $q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_1 \in Q_F$

Positions 0 1 2 3 4

We have $X_0 = \{0\}$, $X_1 = \{1,3\}$, $X_2 = \{2,4\}$.

Construction:

Let $L = L(A)$ with $A = (Q, q_0, \to, Q_F)$.

Wlog, $Q = \{q_0, \dots, q_n\}$.

Then we define

$$\varphi_L := \exists X_0 \dots \exists X_n : (1) \wedge (2) \wedge (3) \wedge 4 \; (\wedge (5) \text{ if } \varepsilon \notin L)$$

with

(1) $\displaystyle\bigwedge_{0 \le i \ne j \le n} \forall x : \neg (X_i(x) \wedge X_j(x))$

(2) $\forall x : \text{first}(x) \to X_0(x)$.

(3) $\forall x : \forall y : \text{succ}(x,y) \to \displaystyle\bigvee_{q_i \xrightarrow{a} q_j} (X_i(x) \wedge P_a(x) \wedge X_j(y))$

(4) $\forall x : \text{last}(x) \to \displaystyle\bigvee_{q_i \xrightarrow{a} q_j \in Q_F} (X_i(x) \wedge P_a(x))$

(5) $\exists x : x = x$.

Intuitively:

(1) Every letter stems from a single state (no branching in the word)

(2) Run starts in q_0

(3) Successor state respects transition relation

(4) Last letter leads to a final state

(5) There is a letter