5

Two-variable logics

5.1 Introduction

In this and subsequent chapters we study two-variable logic for data words. Two-variable logic is the subclass of first-order logic containing formulas which use only two variables $x$ and $y$. Unlike the full first-order logic whose satisfiability and finite satisfiability problems are undecidable, for two-variable logic both these problems are decidable [Mor75]. More precisely they are complete for \textsc{Nexptime} [GKV97]. The expressiveness of this logic is good enough for many applications in AI and natural language processing.

5.2 Preliminaries

In the following, $\mathbb{N}$ denotes the set of natural numbers and $\mathbb{Q}$ the set of rationals. We deal with equivalence relations, preorders and linear orders and briefly introduce them now. Let $A$ be a finite set. An equivalence relation $\sim$ on $A$ is a reflexive, symmetric and transitive relation. A total preorder $\leq_p$ on $A$ is a transitive, reflexive, total relation, that is, $u \leq_p v$ and $v \leq_p w$ implies $u \leq_p w$ and for every two elements $u, v \in A$ $u \leq_p v$ or $v \leq_p u$ holds. A linear order $\leq_l$ on $A$ is a antisymmetric total preorder, that is, if $u \leq_l v$ and $v \leq_l u$ then $u = v$. Thus, the essential difference between a total preorder and a linear order is that the former allows that for two distinct elements $u$ and $v$ both $u \leq_p v$ and $v \leq_p u$ hold. We
call two such elements equivalent with respect to $\leq_p$. Thus, a total preorder can be viewed as an equivalence relation $\sim_p$ whose equivalence classes are linearly ordered by $\leq_p$. Clearly, every linear order is a total preorder with equivalence classes of size one. For any element $u$, the $\sim_p$-class of $u \in A$ is denoted by $[u]_{\sim_p}$ (or $[u]$ if $\sim_p$ is clear from the context). The set of all equivalence classes of $\sim_p$ is denoted by $A/\sim_p$.

We only consider finite structures. Therefore, the linear order on the equivalence classes of a total preorder induces a successor relation of the equivalence classes. We write $+1_p^s(u,v)$ if the equivalence class of $v$ with respect to $\leq_p$ is the successor of the equivalence class of $u$ and we call $+1_p^s$ the induced successor relation of $\leq_p$. Further we say $u$ and $v$ are $+1_p$-close, if either $u+1_p^s v$ or $u \sim_p v$ or $v+1_p^s u$. If $u \leq_p v$ and if they are not $+1_p$-close, we denote it by $u \ll_p v$. Similarly for $+1_l(u,v)$ and $+1_l$-close.

We use binary relation symbols $\leq_{l_1}, \leq_{l_2}, \ldots$ that are always interpreted as linear orders, binary relation symbols $\leq_p, \leq_{p_1}, \leq_{p_2}, \ldots$ that are interpreted as total preorders, and binary relation symbols $+1_p, +1_{p_1}, +1_{p_2}, \ldots$ as well as $+1_l, +1_{l_1}, +1_{l_2}, \ldots$ that are interpreted as successor relations.

A first order structure $\mathfrak{A}$ is a non-empty set $A$ (called the universe) along with some specified binary relations. For example, finite words over the alphabet $\Sigma$ are (usually) represented as first-order structures of the form $([n], (P_a)_{a \in \Sigma}, <, +1)$ where $<$ and $+1$ are the order and successor relations on natural numbers (restricted to the set $[n]$) and $(P_a)_{a \in \Sigma}$ are unary predicates representing the $\Sigma$ labelling on positions. Often while denoting the vocabulary of the structure we abbreviate unary predicates by the alphabet they are representing, for instance $(P_a)_{a \in \Sigma}$ by $\Sigma$.

An ordered structure is a structure with non-empty universe and some linear orders, some total preorders, some successor relations and some unary relations. We always allow an unlimited number of unary relations and specify the numbers of allowed linear orders and total preorders explicitly. For instance, a $(+1_{l_1}; +1_{p_2}, \leq_{p_2})$-structure is a structure with arbitrarily many unary relations, one successor of linear order and one total preorder together with a corresponding successor relation. We write $(+1_{l_1}; +1_p, \leq_p)$ instead of $(+1_{l_1}; +1_{p_2}, \leq_{p_2})$ if no ambiguities arise.
5.2.1 Data words

Given a data word $w$, the data values define an equivalence relation on the positions of $w$ given by $i \sim j$ if $d_i = d_j$. Thus a data word can be naturally represented as a first-order structure $w = ([n], \Sigma, <, +1, \sim)$.

Assume the data alphabet $\Gamma$ is linearly ordered by an order relation $<_\Gamma$. In this case data values $d_i$ and $d_j$ on positions $i$ and $j$ can have any of the following relationships: $d_i = d_j$ or $d_i <_\Gamma d_j$ or $d_i >_\Gamma d_j$. This relationship can be expressed by a total preorder on positions given by,

$$i \leq_p j \iff d_i <_\Gamma d_j \text{ or } d_i = d_j.$$

Hence an ordered data word can be represented logically as a first order structure $w = ([n], \Sigma, \leq_t, +1, \sim)$; where $\leq_t$ denotes the linear order on positions and $\leq_p$ denotes the total preorder on positions induced by the order on the data values.

Note that for a linear order and a total preorder the successor relation uniquely defines the order and vice-versa. Therefore even if one of the successor or order relation is absent from the vocabulary, every (ordered) data word has a unique first-order representation in the above mentioned scheme.

Example 5.2.1. The word $ababab$ is encoded as the structure,

$$(6, P_a = \{1, 3, 5\}, P_b = \{2, 4, 6\}, <, +1).$$

Example 5.2.2. The data word $(a, d_2)(b, d_4)(a, d_1)(b, d_2)(a, d_3)(b, d_2)$ is encoded as the structure,

$$(6, P_a = \{1, 3, 5\}, P_b = \{2, 4, 6\}, <, +1, \sim = \{\{1, 4, 6\}, \{2, 3\}, \{5\}\}).$$

Example 5.2.3. The ordered data word $(a, 1)(b, 2)(a, 1)(b, 4)(a, 2)(b, 1)$ is encoded as the structure,

$$(6, P_a = \{1, 3, 5\}, P_b = \{2, 4, 6\}, <, +1, \leq_p).$$
where \( \leq_p \) is the total preorder \( \{1, 3, 6\} \leq_p \{2, 5\} \leq_p \{4\} \).

### 5.3 Logics

The set of first order (abbreviated as FO) formulas over the vocabulary \( \tau \) is given by the following syntax:

\[
\varphi ::= x = y \mid R(x_1, \ldots, x_n) \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \neg \varphi \mid \exists x \varphi
\]

where \( R \) is an \( n \)-ary relation as specified by \( \tau \) and \( x, y, x_1 \ldots \) are first-order variables. The set of monadic second order (abbreviated as MSO) formulas over the vocabulary \( \tau \) is given by the syntax

\[
\varphi ::= x = y \mid R(x_1, \ldots, x_n) \mid X(x) \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \neg \varphi \mid \exists x \varphi \mid \exists X \varphi
\]

where \( X \) is a set variable. Note that in MSO variables \( X_1, X_2, \ldots \) range over subsets of the universe. Two-variable first-order logic or simply Two-variable logic is the restriction of first order logic to formulas that only use (at most) two variables \( x \) and \( y \). We denote two-variable logic by \( \text{FO}^2 \). Similarly the three-variable logic is denoted by \( \text{FO}^3 \). Formulas with no free variables are called sentences, but in the following we may refer to sentences as formulas when no ambiguity arises.

It is not possible to express in \( \text{FO}^2 \) that a binary relation \( R \) is transitive, a fact easily proved by EF-games. Hence we need to supply the logic with additional non-logical symbols if some relations are to be interpreted as order or equivalence relations. These are specified in the vocabulary. For instance \( \text{FO}^2 (\Sigma, <, +1) \) is the two variable logic with unary predicates and binary relations \( <, +1 \) interpreted as a linear order and its successor relation. In other words, this is the two-variable logic on words.

**Example 5.3.1.** The following \( \text{FO}^2 (\Sigma, <, +1) \) formula describes that the model (in this case a word) contains three ‘a’s.

\[
\varphi_1 = \exists x (P_a(x) \land \exists y (x < y \land P_a(y) \land \exists x (y < x \land P_a(x)))).
\]
Example 5.3.2. The following $\text{FO}^2(\Sigma, <, +1)$ formula says that the word is from the language $a^*b^*$.

$$\varphi_2 = \forall x \forall y (P_a(x) \land P_b(y) \rightarrow x < y).$$

Example 5.3.3. The following $\text{FO}^3(\Sigma, <, +1, \sim)$ formula over data words describes that between any two positions of the same class there is no ’b’-labelled position from a different class.

$$\varphi_3 = \forall x \forall y \forall z (x \sim y \land P_b(z) \land x < z \land z < y \rightarrow z \sim x).$$

Example 5.3.4. The formula below states that each class contains an ‘a’ if it contains a ‘b’ and vice versa.

$$\varphi_4 = \forall x ((P_a(x) \rightarrow \exists y (P_b(y) \land x \sim y)) \land (P_b(x) \rightarrow \exists y (P_a(y) \land x \sim y))).$$

Example 5.3.5. The following $\text{FO}^2(\Sigma, <, +1, \leq_p)$ formula over ordered data words describes that the data values on the positions are non-decreasing.

$$\varphi_4 = \forall x \forall y (x < y \rightarrow x \leq_p y).$$

5.3.1 Scott reduction

A very useful property of $\text{FO}^2$ formulas is that they possess a normal form, called Scott Normal Form, with quantifier rank at most two. The following fact is due to Dana Scott [Sco62]. Fix a relational vocabulary $\tau$ containing order relations. A formula $\varphi \in \text{FO}^2(\tau)$ is equivalent with respect to satisfiability (as well as finite satisfiability) to a formula of the form;

$$\zeta = \forall x \forall y \chi \land \bigwedge_{i=1}^{i=k} \forall x \exists y \psi_i,$$

where $k \in \mathbb{N}$ and, $\chi$ and $\psi_i$ are quantifier free formulas which use only extra unary predicates other than the predicates used in $\varphi$. The formula $\zeta$ can be obtained from $\varphi$ in linear time and the size of the formula $\zeta$ is linear in terms of the size of $\varphi$. Moreover, models of $\zeta$ are expansions of models of $\varphi$ with unary predicates.
Chapter 5. Two-variable logics

and models of \( \varphi \) are reducts of models of \( \zeta \). A full proof of the above statement can be found in [GKV97].

5.4 FO\(^2\) on data words

The primary reason why two-variable logics are looked at in the context of data words is stated below.

**Theorem 5.4.1.** Finite satisfiability problem of \( \text{FO} (\Sigma, \leq_l, +1_l, \sim_p) \) is undecidable. More precisely, finite satisfiability problem of \( \text{FO}^3 (\Sigma, \leq_l, +1_l, \sim_p) \) is undecidable.

The above theorem was proved in [BDM+11] which also showed the landmark result that;

**Theorem 5.4.2.** Finite satisfiability problem of \( \text{FO}^2 (\Sigma, \leq_l, +1_l, \sim_p) \) is decidable and is as hard as reachability of multicounter automata.

The proof of the above theorem is via automata construction and is interesting in many aspects. Given a formula \( \varphi \in \text{FO}^2 (\Sigma, \leq_l, +1_l, \sim_p) \) it is converted in 2-DEXPTIME to a Data automaton \( A_\varphi \) such that \( L(\varphi) = L(A_\varphi) \). Since checking nonemptiness of Data automaton is decidable it implies that checking (finite) satisfiability of \( \text{FO}^2 (\Sigma, \leq_l, +1_l, \sim_p) \) is decidable. But the complexity of this decision procedure as stated above is as hard as the reachability problem of multicounter automata which is not known to be elementary, making it untenable for practical applications. On the other hand since classical logics provides tools and techniques to test and compare expressiveness questions, this result has great importance.

The proof in [BDM+11] also shows that Data automata are characterized by the logic \( \text{EMSO}^2 (\Sigma, \leq_l, +1_l, \sim_p, \oplus 1) \) whose formulas are of the form \( \exists X_1 \ldots X_n \varphi \) where \( \varphi \in \text{FO}^2 (\Sigma, \leq_l, +1_l, \sim_p, \oplus 1) \) and \( X_1, \ldots, X_n \) are set variables. A merit of this proof method is that it allows us to prove decidability without proving a small model property. Note that in this case an elementary small-model property will settle a decades-old problem (is reachability problem for Petri nets elementarily decidable?). In the next two chapters we will emulate this proof method (the history of which dates back to Büchi) to show decidability of other logics.
Next we move on to ordered data words. As mentioned earlier a linear order on data values will imply a total preorder on the positions of the data word. Hence two-variable logic on ordered data words has the signature $\text{FO}^2(\Sigma, \leq_l, +1_l, \preceq_p)$.

The following was proved in [BDM+11]:

**Theorem 5.4.3.** Finite satisfiability problem of $\text{FO}^2(\Sigma, \leq_l, +1_l, \preceq_p)$ is undecidable.

Even if we replace the preorder $\preceq_p$ with its successor relation $+1_p$ the undecidability remains as is shown below.

**Theorem 5.4.4.** Finite satisfiability problem of $\text{FO}^2(\Sigma, \leq_l, +1_l, +1_p)$ is undecidable.

**Proof.** The proof follows the lines of the proof of Proposition 29 in [BDM+11].

We reduce from the Post’s Correspondence Problem. Let $I = (u_1, v_1), \ldots, (u_k, v_k)$ be an instance of PCP. We construct an $\text{FO}^2(\leq_l, +1_l; +1_p)$-sentence $\varphi$ that has a finite model if and only if $I$ has a solution. The sentence $\varphi$ uses unary predicates from $\Sigma$ as well as the two unary predicates $U, V$, and expresses the following conditions:

1. The string projection of $\leq_l$ is $u_1v_1 \ldots u_mv_m$ for some $m \in \mathbb{N}$. Elements corresponding to some $u_i$ and $v_i$ are marked with $U$ and $V$, respectively.

2. Every equivalence class of $+1_p$ contains exactly two elements such that
   - One is marked with $U$ and one is marked with $V$.
   - Both carry the same label from $\Sigma$.

3. Positions $x_1, \ldots, x_{|u|}$ corresponding to the positions of $u := u_1 \ldots u_m$ fulfill $+1_p(i, i+1)$ for all $i \in \{1, \ldots, |u| - 1\}$. Analogously for $v$.

Condition (1) can be expressed in the following way. Given a string $u_iv_i$, it is straightforward to write a formula $\varphi_{u_iv_i}(x) \in \text{FO}^2(\Sigma, +1_l)$ which states that there is a subword $u_i v_i$ starting from the position $x$ where positions of $u_i$ are labelled by $U$ and positions of $v_i$ are labelled by $V$. In addition, the subword is followed by a $U$ position unless the word ends. Next we state that;
∀x \left( U(x) \land (\exists y \ (y + 1(y, x) \land V(y)) \lor \neg \exists y \ + 1(y, x)) \rightarrow \bigvee_{i \in k} \varphi_{u_iV_i}(x) \right)

The second condition is ensured by the formulas;

\neg \exists x \exists y (x \sim_p y \land x \neq y \land ((U(x) \land U(y)) \lor (V(x) \land V(y))))

\forall x \bigwedge_{a \in \Sigma} (P_a(x) \land U(x) \rightarrow \exists y \ (P_a(y) \land x \sim_p y \land V(y)))

\forall x \bigwedge_{a \in \Sigma} (P_a(x) \land V(x) \rightarrow \exists y \ (P_a(y) \land x \sim_p y \land U(y)))

The third condition can be ensured by the formula,

∀x∀y(U(x) \land U(y) \land +1_p(x, y) \rightarrow x <_l y)

Now, from a solution \vec{i} = i_1 \ldots i_m a model of \varphi can be constructed easily. On the other hand, let \mathcal{M} be a model of \varphi. By (1), the string projection of \mathcal{M} is of the form u_{i_1}v_{i_1} \ldots u_{i_m}v_{i_m}. The U- and V-labeled elements are ordered with respect to \leq_p due to (3). Thus, (2) implies that u_{i_1} \ldots u_{i_m} = v_{i_1} \ldots v_{i_m}.

This means that for two-variable logic to be decidable on ordered data words either the linear order \leq_l or the successor relation +1_l has to be dropped from the vocabulary. Following this line in [SZ10, SZ11] it was shown that,

**Theorem 5.4.5.** Finite satisfiability problem of \text{FO}^2(\Sigma, \leq_l, \leq_p, +1_p) is decidable in \text{EXPSPACE}.

The above theorem is proved by showing a small model property. In the subsequent chapters we consider the other line that is to drop \leq_l. The status of finite satisfiability problem for \text{FO}^2(\Sigma, +1_l, \leq_p, +1_p) is still open. In the next chapter we restrict the preorder to be a linear order and study the logic with two linear orders, namely \text{FO}^2(\Sigma, \leq_{l_1}, +1_{l_1}, +1_{l_2}) and its subclasses. While this is the two-variable logic on class of ordered data words where all data values appearing in the word are different, this logic is interesting in its own way as described in the next paragraph.
The status of satisfiability problem of first-order logic on ordered structures is very interesting as these are one of the simplest mathematical structures and at the same time ubiquitous in computer science as they naturally arise in computation. To give a short account of the results in this direction, in [EVW02] it is shown that the satisfiability and finite satisfiability problems of $\text{FO}^2$ over words are NExptime-complete. In [Ott01] the following are shown. The logic $\text{FO}^2$ over ordered or well-ordered domains, or in the presence of one well-founded relation, is decidable for satisfiability as well as for finite satisfiability. The complexity of these decision problems is essentially the same as for plain unconstrained $\text{FO}^2$. In contrast, $\text{FO}^2$ becomes undecidable for satisfiability and for finite-satisfiability, if a sufficiently large number of predicates (at least eight) are required to be interpreted as orderings, well-orderings, or as arbitrary well-founded relations. In [KO05] it is shown that $\text{FO}^2$ with two transitive relations (without equality) is undecidable. In [KO05] it is shown that $\text{FO}^2$ is undecidable with three equivalence relations, but is decidable when the number of equivalence relations is two. Later in [KT09] it is shown that in the case of two equivalence relations, finite satisfiability is decidable in 3-Exptime. In the same paper the undecidability is sharpened to one equivalence relation and one transitive relation.

As a warm-up, we show the following theorem. Note that $+2_{l_1}$ denotes the second-successor or successor-of-successor relation in the linear order $\leq_{l_1}$. Similarly for $+3_{l_1}$.

**Theorem 5.4.6.** The finite satisfiability problems for the following logics are undecidable.

(a) $\text{FO}^2 (\Sigma, \leq_{l_1}, +1_{l_1}, \leq_{l_2}, +1_{l_2})$

(b) $\text{FO}^3 (\Sigma, +1_{l_1}, +1_{l_2})$

(c) $\text{FO}^2 (\Sigma, +1_{l_1}, +2_{l_1}, +3_{l_1}, +1_{l_2}, +2_{l_2})$

**Proof.** We reduce the Post’s Correspondence Problem (PCP) to the finite satisfiability problems of the logics $\text{FO}^2 (\Sigma, +1_{l_1}, \leq_{l_1}, +1_{l_2}, \leq_{l_2})$, $\text{FO}^3 (\Sigma, +1_{l_1}, +1_{l_2})$ and $\text{FO}^2 (\Sigma, +1_{l_1}, +2_{l_1}, +3_{l_1}, +1_{l_2}, +2_{l_2})$. The variant of PCP in which the strings are of length one or two is also undecidable [HU79]. We employ this variant for the reduction. Assume that we are given a PCP instance $I = \{(u_i, v_i) \mid i \in [n], u_i, v_i \in \Sigma^{\leq 2}\}$
Chapter 5. Two-variable logics

over the alphabet $\Sigma = \{l_1, l_2, \ldots, l_k\}$. We encode the PCP solution as structures in the above vocabularies, in the following way. Let $\Sigma' = \{l'_1, l'_2, \ldots, l'_k\}$ and $\hat{\Sigma} = \Sigma \cup \Sigma'$. Given a word $w = a_1a_2 \ldots a_n$ in $\Sigma^*$, we denote by $w'$ the word $a'_1a'_2 \ldots a'_n$ in $\Sigma'^*$.

A solution of $I$ is a structure $\mathcal{A} = (A, \hat{\Sigma}, +1_{l_1}, +1_{l_2})$ over $\hat{\Sigma}$ such that,

1. The word $(A, \hat{\Sigma}, +1_{l_1})$ is in the language $(u_1v'_1 + u_2v'_2 + \ldots + u_nv'_n)^+$. This language is expressible in $\text{FO}^2 (\hat{\Sigma}, +1_{l_1})$ as in the proof of Theorem 5.4.4, let us call it $\varphi_1$.

2. The word $(A, \hat{\Sigma}, +1_{l_2})$ is in the language $(l_1l'_1 + l_2l'_2 + \ldots + l_kl'_k)^+$. This language is expressible in $\text{FO}^2 (\hat{\Sigma}, +1_{l_2})$ by the formulas (call them $\varphi_2$),

\[
\forall x \forall y \left( \bigwedge_i (P_i(x) \land +1_{l_2}(x, y) \rightarrow P'_i(y)) \land \bigwedge_i (P'_i(x) \land +1_{l_2}(x, y) \rightarrow P_i(y)) \right)
\]

\[
\exists x \left( \neg (\exists y +1_{l_2}(y, x)) \rightarrow \bigvee_i P_i(x) \right) \land \exists x \left( \neg (\exists y +1_{l_2}(y, x)) \rightarrow \bigvee_i P'_i(x) \right)
\]

(3a) The third condition is specific for each of the logics, though they all express the same form of matching between $\Sigma$ and $\Sigma'$ positions. We say $x$ is $\Sigma$-position, denoted as $\Sigma(x)$, if it is labeled by a letter from $\Sigma$, that is if $P_i(x) \lor P'_i(x)$ is true. Similarly, we say $x$ is a $\Sigma'$-position, denoted as $\Sigma'(x)$, if $P'_i(x) \lor P'_i(x)$ is true. Our next condition says that, taken only the $\Sigma$ positions, the order $\leq_{l_1}$ respects the order $\leq_{l_2}$, similarly is the case with $\Sigma'$ positions. This can be expressed by the following formula in $\text{FO}^2 (\hat{\Sigma}, +1_{l_1}, \leq_{l_1}, +1_{l_2}, \leq_{l_2})$,

\[
\varphi_{3a} \equiv \forall xy \ ((\Sigma(x) \land \Sigma(y) \land x \leq_{l_1} y ) \rightarrow x \leq_{l_2} y)
\]

\[
\land ((\Sigma'(x) \land \Sigma'(y) \land x \leq_{l_1} y ) \rightarrow x \leq_{l_2} y))
\]

(3b) Let $S(x, y)$ be true if either one of the following conditions holds : (1) both $x$ and $y$ are $\Sigma$ positions and no position between $x$ and $y$ in $+1_{l_1}$ is labeled from $\Sigma$. (2) Analogously, both $x$ and $y$ are $\Sigma'$ positions and no position between $x$ and $y$ in $+1_{l_1}$ is labeled from $\Sigma'$. Notice that $S(x, y)$ can be
coded in FO³(Σ, +1l₁, +1l₂) since the distance between any two consecutive Σ positions or any two consecutive Σ′ positions is bounded by two. The formula $S(x, y) = S_Σ(x, y) \lor S_{Σ'}(x, y)$. Below we give the definition of $S_Σ(x, y)$ while $S_{Σ'}(x, y)$ is defined analogously.

$S_Σ(x, y) = (Σ(x) \land Σ(y)) \land$

$(+1l₁(x, y)$

$\lor \exists z (+1l₁(x, z) \land Σ'(z) \land +1l₁(z, y))$

$\lor \exists z (+1l₁(x, z) \land Σ'(z) \land \exists x (+1l₁(z, x) \land Σ'(x) \land +1l₁(x, y))))$

Once we have $S$ we enforce the correct matching in the following way, $ϕ_{3b}$ is the conjunction of the following formulas in FO³(Σ, +1l₁, +1l₂),

$∀xyz((Σ(x)\landΣ(y)\landΣ'(z)\land S(x, y) \land x + 1l₂z) \rightarrow z + 1l₂y)$

$∀xyz((Σ'(x)\landΣ'(y)\landΣ(z)\land S(x, y) \land x + 1l₂z) \rightarrow z + 1l₂y)$

(3c) Note that, when the strings are of length at most two, the predicate $S(x, y)$ defined above, can be coded by using the successor relations $+1l₁, +2l₁$ and $+3l₁$ as in the previous case. Again, we define $S(x, y) = S_Σ(x, y) \lor S_{Σ'}(x, y)$ and $S_Σ(x, y)$ is;

$S_Σ(x, y) = (Σ(x) \land Σ(y)) \land$

$(+1l₁(x, y)$

$\lor (+2l₁(x, y) \land \exists y (+1l₁(x, y) \land Σ'(y))))$

$\lor (+3l₁(x, y) \land \exists y (+2l₁(x, y) \land Σ'(y)) \land \exists y (+1l₁(x, y) \land Σ'(y))))$

The matching is done by $ϕ_{3c}$ which is a conjunction of the following formulas in $FO²(Σ, +1l₁, +2l₁, +3l₁, +1l₂, +2l₂)$,

$∀xy ((Σ(x) \land Σ(y) \land S(x, y)) \rightarrow x + 2l₂y)$

$∀xy ((Σ'(x) \land Σ'(y) \land S(x, y)) \rightarrow x + 2l₂y)$

We claim that the formulas $ϕ_1 \land ϕ_2 \land ϕ_{3a}, ϕ_1 \land ϕ_2 \land ϕ_{3b}, ϕ_1 \land ϕ_2 \land ϕ_{3c}$ encodes
a solution of $I$ in the logics $\text{FO}^2\left(\hat{\Sigma}, +1_{l_1}, \leq_{l_1}, \leq_{l_2}\right)$, $\text{FO}^3\left(\hat{\Sigma}, +1_{l_1}, +1_{l_2}\right)$, $\text{FO}^2\left(\hat{\Sigma}, +1_{l_1}, +2_{l_1}, +3_{l_1}, +1_{l_2}, +2_{l_2}\right)$ respectively. That is $I$ has a solution if and only if each of them is satisfiable. Suppose $I$ has a solution $i_0, i_1, \ldots, i_m$, in which case $u_{i_0} u_{i_1} \ldots u_{i_m} = v_{i_0} v_{i_1} \ldots v_{i_m}$, call it $w$. Let $|w| = n$. We define the structure $\left([2n], \hat{\Sigma}, +1, +1_{l_2}\right)$ such that $+1$ is the successor relation on $[2n]$ and $\left([2n], \hat{\Sigma}, +1\right)$ is the word $u_{i_0} v'_{i_0} \ldots u_{i_m} v'_{i_m}$. Note that in this word there are $n$-many $\Sigma$ positions and $\Sigma'$ positions. Let those be the sequences $\sigma_1 \ldots \sigma_n$ and $\sigma'_1 \ldots \sigma'_n$ in the ascending order. Define the order $+1_{l_2}$ as $\sigma_1 \sigma'_1 \sigma_2 \sigma'_2 \ldots \sigma_n \sigma'_n$. Clearly the structure satisfies all the three conditions. Now suppose a structure satisfies all the three conditions. Without loss of generality we can assume that it is of the form $\left([2n], \hat{\Sigma}, +1, +1_{l_2}\right)$ for some $n \in \mathbb{N}$ such that $\left([2n], \hat{\Sigma}, +1\right)$ is a word of the form $u_{i_0} v'_{i_0} \ldots u_{i_m} v'_{i_m}$ for some $i_0 \ldots i_m$. Let $\sigma_1 \ldots \sigma_n$ and $\sigma'_1 \ldots \sigma'_n$ be the $\Sigma$ and $\Sigma'$ positions in the ascending order. Condition (3) ensures that for every $i$, $\sigma_i + 1_{l_2} \sigma'_i + 1_{l_2} \sigma_{i+1}$ (if $\sigma_{i+1}$ exists) and condition (2) ensures that $\sigma_i$ is labelled by letter ‘$l$’ if and only if $\sigma_i$ is labelled by ‘$l'$’. Together it implies that $u_{i_0} u_{i_1} \ldots u_{i_m} = v_{i_0} v_{i_1} \ldots v_{i_m}$. 

Note that undecidability of $\text{FO}^3\left(\Sigma, +1_{l_1}, +1_{l_2}\right)$ also implies undecidability of $\text{FO}^3\left(\Sigma, \leq_{l_1}, \leq_{l_2}\right)$ since in three variables the successor relation $+1_{l_1}$ is expressible in terms of the order relation $\leq_{l_1}$. An interesting question is to sharpen the undecidability of $\text{FO}^2\left(\Sigma, +1_{l_1}, +2_{l_1}, +3_{l_1}, +1_{l_2}, +2_{l_2}\right)$ by reducing the number of successors required. In the next chapter we will show that $\text{FO}^2\left(\Sigma, +1_{l_1}, +1_{l_2}\right)$ is decidable.