0 Announcements

- The Project Proposal is due Friday 03/04. Those who have not discussed with Professor Kfoury about their project, please do so as soon as possible. In the Project Proposal you are expected to present a road map of the steps you are planning to follow in order to complete the project.

1 Predicate Logic: Definability and Expressiveness

In the previous lectures we established that reachability in graphs is not a first-order property, and therefore it could not be modeled. In this section we are going to show the proof presented in page 4 of Handout 12 and in page 138 [LCS]. More specifically, we are going to prove the following:

**Theorem:** Let $\psi$ be a first-order sentence such that, for every $n \geq 1$, there is a model of $\psi$ with at least $n$ elements.

**Conclusion:** $\psi$ has an infinite model

**Proof:**

**Step 1:** Let for every $n \geq 1$, be $\theta_n \triangleq \exists x_1 \exists x_2 \exists x_3 ... \exists x_n \land \neg (x_i = x_j)$. This sentence expresses the fact that there exist elements in the universe, whose number is up to $n$, such that all the pairs of inequality are satisfiable, i.e. there exist at least $n$ distinct elements. The negation guarantees that no two elements are the same.

**Step 2:** $\Delta \triangleq \{\psi\} \cup \{\theta_n | n \geq 1\}$

In the second step of the proof we have presumed the existence of $\psi$, which assumes that for every finite structure there exists a model for it. In other words the above is a claim that $\psi$ satisfies every finite size.

**Step 3:** Every finite subset $\Delta'$ of $\Delta$ is satisfiable, and at this point we are going to explain why this is true. Consider any infinite subset of $\Delta$, and then include $\psi$. Then, $\psi$ will introduce a number of $\theta_i$, where $1 \leq i \leq n$. For the purpose of our example we assume that these $\theta$'s are the following: ($\theta_1, \theta_2, ..., \theta_{52}$). Given that the collection is finite, there exists a maximum index within this collection which in this case is 52. Thus, the largest element index in the specific collection will be satisfied by a model with 53 elements. Given that the collection is infinite, we can then assume that a model exists for $\psi$ and that $\theta$ is defined. For every finite subset of $\Delta$ that is consistent (satisfiable), then the whole $\Delta$ is consistent (satisfiable).

**Step 4:** By compactness, the set $\Delta$ is satisfiable.

**Step 5:** Any model of $\Delta$ must be infinite and this can be shown using an argument by contradiction. Consider a model that is not infinite, and that contains 52 elements. Then $\Delta$ should contain $\theta_n$, where $n = 53$. Yet, $\theta_{53}$ assumes at least 53 elements so it is a contradiction to say that $\Delta$ contains 52 elements since it should be true for all $\theta$. The corollary proof that we presented claims that there are 52 elements, i.e. $\theta_{52}$ and that $\Delta$ assumes the existence of all $\theta$. So for $\theta_{53}$ there is a contradiction.

**Löwenheim-Skolem Theorem**

There is no first-order sentence $\psi$, such that it is satisfiable by all finite structures and by no infinite structure. In particular, there cannot exist a $\psi$ that is true in some model structure and not in the infinite model. Another way to express the above statement is that finiteness of structures is not a first-order property. Note that another use of compactness is topological sorting. In general, the methodology is to first prove a fact and then extend it using compactness.
2   Extended Example in First Order Logic

The material for this section is related to Handout 14 and its main focus is on what expressions can be derived from first-order models. More specifically, if $M$ is a model that belongs to the first-order theory of $M$ then it is:

$$Th(M) \equiv \{ \phi \mid \phi \text{ is a 1st-order sentence s.t. } M \models \phi \}$$

Possible structures are shown in page 10 of Handout 14. More specifically, let us study the model (signature) $N \equiv (\mathbb{N}, 0, S)$ which comprises the following elements:

- $\mathbb{N}$: Set of natural numbers
- 0: Constant zero
- $S$: Successor function

Note that there are no symbols for the numbers 1, 2, ..., $n$ in the signature. However, these numbers can be constructed using the macro of 0. For example, the number 1 can be constructed by the macro $S(0)$. Similarly the number 2, can be constructed as a macro of 1, i.e. $S(S(0))$ and so on. Recall that $S$ is a unary function that represents the successor function. Thus, it maps the number 0 to the number 1, the number 1 to the number 2, the number 2 to the number 3 and so on. We stress out that there is an infinite number of possible sentences that can be extracted from the specific signature. Some of the first-order properties (facts) that we can state for this structure are the following:

$$S1 \forall x \neg (Sx = 0)$$

$$S2 \forall x \forall y (Sx = Sy \rightarrow x = y): \text{ This states that for every two elements, if the successor of } x \text{ is equal to the successor of } y, \text{ then } x \text{ and } y \text{ are the same element. The existence of equal images, leads to the}\ \text{conclusion that the successive function is an injective function, i.e. a one-to-one function. Recall that } S \text{ is a unary function that represents the success}\ \text{or function. Thus, it maps the number 0 to the number 1, the number 1 to the number 2, the number 2 to the number 3 and so on. We stress out that there is an infinite number of possible sentences that can be extracted from the specific signature. Some of the first-order properties (facts) that we can state for this structure are the following:}$$

$$S3 \forall y (\neg (y = 0) \rightarrow \exists x (y = Sx)): \text{ This states that for every element that is not 0, there exists an } x \text{ such that it is the pre-image of } y. \text{ Note that 0 is the only number that does not have a predecessor, while every other element has a predecessor.}$$

$$S4.1 \forall x \neg (Sx = x)$$

$$S4.2 \forall x \neg (SSx = x)$$

$$\ldots$$

$$S4.n \forall x \neg (\underbrace{SS\ldots S}_{n}x = x)$$

$$\ldots$$

Properties $S4.1 - S4.n - \ldots$ prevent the existence of cycles, i.e. guarantee that there are no loops. This is explicitly defined with the use of the negation.

Let us consider the above facts to be premises. We will focus on what can we model in first-order logic based on this structure. In other words we are aiming at answering the question: "What can we deduce/extract out of these sentences?"

The answer is that any possible first-order fact that can be modeled, is also formally deducible by the given structures $S1,S2,S3,S4.1...S4.n...$ This is formally stated in page 11 of Handout 14.

- Let $\Gamma = \{ S1, S2, S3, S4.1, S4.2, S4.3... \}$
Clearly $N \models \phi$ for every $\phi \in \Gamma$ so that $\Gamma \subseteq \text{Th}(N)$.

The question is what can we say about $\bar{\Gamma} = \{ \phi \mid \phi \text{ 1st order sentence s.t. } \Gamma \vdash \phi \}$?

Note that the bar on $\Gamma$ shows the deductive closure of the sentences $\{S1, S2, S3, S4.1, S4.2, S4.3\ldots\}$.

The answer is that we can exhaust the first-order facts using the sentences of $\Gamma$. More specifically, $\Gamma$ is an axiomatization of $\text{Th}(N)$ because it is a collection of sentences from which every sentence $\phi$ made true by $N$, can be formally deduced.

Furthermore, FACT 1 presented in page 12 of Handout 14 states that the first-order theory of each $N, N_1$ and $N_2$ can be extended, so that it is not only axiomatizable but also decidable. Recall that a problem is decidable if there is an algorithm to check whether or not the input has some property. We stress out that decidability does not represent how easy it is to decide this property, like complexity does, as it may take a very long time to decide some property. At this point we introduce the reader to the notion of semi-decidable algorithms by presenting an example. Note that this is general, but important knowledge for computer scientists.

**General knowledge:** A semi-definite algorithm generates all the numbers of a set, but it runs forever. For example, in the case of a Turing machine there is an infinite tape that runs forever and every small time intervals the machine generates a number on the tape. The list of numbers that the machine will produce if it runs forever, is the set. Let us assume that we have a specific number, for example 32, and we want to study if it belongs to the set. If this number belongs to the set, then we might have to wait for a very long time but the number will eventually appear. Then we can guarantee that this number is an element of the set. However, if the number 32 is not a member of the set then we will have to wait forever and still not have any guarantees about whether the element belongs to the set or not. Thus, in the case of semi-decidability, we are certain about decidability but the same does not apply for undecidability. Recall that undecidability is a decision problem for which it is known to be impossible to construct an algorithmic solution. To sum up, in the case of a decidable algorithm we can certify that an element is in the set if we wait for a finite amount of time. Furthermore, in a semi-decidable algorithm we can decide membership but not no membership, i.e. we cannot confirm that an element is not in the set. Note that a set can be semi-decidable without being decidable.

An example that illustrates the notion of semi-decidability is the following. Assume the set of all natural numbers $\mathbb{N}$ and two subsets of $\mathbb{N}$, $A \subseteq \mathbb{N}$ and the complement of $A$, $\mathbb{N} - A \subseteq \mathbb{N}$. Then, if there is an algorithm that generates all elements of $A$ and another that produces the elements of its complement, then we have two separate semi-definite algorithms. However, if we combine these two individual algorithms into a third one, then this final algorithm is a decidable one.

### 3 Second-Order Logic

The material of this section is related to Handout 15. Recall from the previous lecture that as we increase the power of what can be modeled, the further away we get from automation.

In pages 7,8 of Handout 15 some examples of second-order logic modeling are presented.

**Well-ordering:** A well-ordering is an ordering $\leq$ such that every non-empty set has a least element with respect to $\leq$:

$$\phi \equiv \forall X[\exists y X(y) \rightarrow \exists v(X(v) \land \forall w(X(w) \rightarrow v \leq w))]$$

The upper case letter $X$ denotes a second-order variable, while the lower case letters represent first-order variables. Note that a well-ordering, is a total order but not a dense order. A dense order is a partial order or a total order $<$ on a set $A$, such that for all $x, y \in A$ for which $x < y$, there exists a $z$ for which $x < z < y$. Intuitively, in the case of a dense order you can always "squeeze" an element between two other elements, like in the case of rational numbers, but that does not apply for the set of integer numbers. For the definition of total ordering we refer the reader to Handout 6. It is important to understand that well-ordering is modeled in the second-order and is not a first-order property. At this point we will consider some examples that belong to the family of well-orderings and one that does
not:

**Well-ordering:** For the set of natural numbers \( \mathbb{N} \), a line can be constructed that contains all the numbers of the set in a sorted manner from the smallest element (0) to the biggest. Note, that this ordering is total, but not dense since we cannot “squeeze” any number in between. Then, consider any subset of this ordering. You will notice that this subset is non-empty and that it contains an element smaller than all the other elements within the subset, i.e. the least number in the subset. For example, if we consider the subset that contains the numbers from 3 until 1000, then the least element will be 3.

**Well-ordering:** Briefly, another example of a well-ordering is the lexicographic ordering, such that:

\[
\{(0, i) | i \in \mathbb{N}\} \cup \{(1, j) | j \in \mathbb{N}\}
\]

Then, \( (0,2215) < (1,22) \) because the first index is smaller that the second.

**Not a well-ordering:** Assume the set of all integers \( \mathbb{Z} \) and again place the elements in a line based on their order from the smallest to the biggest element. Notice, that for the subset with maximum element 3 and all the numbers that precede 3, there is no least number, and therefore the set \( \mathbb{Z} \) is not considered a well-ordering.

- **Finiteness:** In the previous lectures we showed that finiteness is not a first-order property. However, it is a second-order property. More specifically, a second-order sentence satisfied by a model \( M \) if and only if the domain of \( M \) is infinite:

\[
\psi \equiv \exists F[\forall x \forall y(F(x) = F(y) \rightarrow x = y) \land \exists z \forall x \neg(F(x) = z)]
\]

Note that \( \psi \) characterizes a sentence that is infinite, i.e. there exists an \( F \) (in the second-order) such that for every two elements of the domain, \( F \) is a one-to-one (injective) function. Being a unary injective but not surjective function can only happen in an infinite set. Recall that an example of such a unary function is the successive function, where every element has a unique image.

A second-order sentence satisfied by a model \( M \) if and only if the domain of \( M \) is finite:

\[
\neg \psi
\]

Now, the expression of the negation of \( \psi \) characterizes finiteness.

Furthermore, compactness of first-order logic cannot happen in second-order logic, since \( \neg \psi \) characterizes finiteness and \( \theta_1, \theta_2, \ldots, \theta_n \) defines a model to be infinite.

Finally, we stress out that the second-order logic is beyond both, soundness and completeness.