



CS 228 : Logic in Computer Science

Krishna. S

Satisfaction, Validity

- ▶ Given a FO formula $\varphi(x_1, \dots, x_n)$ over a signature τ , is it satisfiable/valid?
 - ▶ Satisfiable, if there exists a τ -structure \mathcal{A} and an assignment α for x_1, \dots, x_n in $u(\mathcal{A})$ such that $\mathcal{A} \models_{\alpha} \varphi(x_1, \dots, x_n)$

Satisfaction, Validity

- ▶ Given a FO formula $\varphi(x_1, \dots, x_n)$ over a signature τ , is it satisfiable/valid?
 - ▶ Satisfiable, if there exists a τ -structure \mathcal{A} and an assignment α for x_1, \dots, x_n in $u(\mathcal{A})$ such that $\mathcal{A} \models_{\alpha} \varphi(x_1, \dots, x_n)$
 - ▶ Valid, if for any τ -structure \mathcal{A} and any assignment α for x_1, \dots, x_n in $u(\mathcal{A})$, $\mathcal{A} \models_{\alpha} \varphi(x_1, \dots, x_n)$
- ▶ Assume we fix the type of the structure \mathcal{A} , say words (why words?)

FOL over Words

- ▶ Given an FO sentence φ over words, is it satisfiable/valid?
- ▶ Satisfiable (Valid) iff some word (all words) satisfies φ

FOL over Words

- ▶ Given an FO sentence φ over words, is it satisfiable/valid?
- ▶ Satisfiable (Valid) iff some word (all words) satisfies φ
- ▶ There could be infinitely many words w satisfying φ
- ▶ $L(\varphi) = \{ \text{words } w \mid w \models \varphi \}$ is called the language of φ

FOL over Words

- ▶ Given an FO sentence φ over words, is it satisfiable/valid?
- ▶ Satisfiable (Valid) iff some word (all words) satisfies φ
- ▶ There could be infinitely many words w satisfying φ
- ▶ $L(\varphi) = \{ \text{words } w \mid w \models \varphi \}$ is called the language of φ
- ▶ Given φ , write an algorithm to check $L(\varphi) = \emptyset$ **Unsatisfiability**.
words which satisfies the FOL formula PHI are the language of the PHI.

First-Order Logic over Words

Expressiveness and Satisfiability

- ▶ Signature for words : $<$, S and Q_a for finitely many symbols a
- ▶ Given a FO formula over words, the signature is fixed

Expressiveness and Satisfiability

- ▶ Signature for words : $<$, S and Q_a for finitely many symbols a
- ▶ Given a FO formula over words, the signature is fixed
- ▶ **Expressiveness**
 - ▶ Given a set of words or a language L , **can you** write a FO formula φ such that $L(\varphi) = L$

Expressiveness and Satisfiability

- ▶ Signature for words : $<$, S and Q_a for finitely many symbols a
- ▶ Given a FO formula over words, the signature is fixed
- ▶ **Expressiveness**
 - ▶ Given a set of words or a language L , **can you** write a FO formula φ such that $L(\varphi) = L$
 - ▶ If you can, FO is expressive enough to capture your language/specification/property

Expressiveness and Satisfiability

- ▶ Signature for words : $<$, S and Q_a for finitely many symbols a
- ▶ Given a FO formula over words, the signature is fixed
- ▶ **Expressiveness**
 - ▶ Given a set of words or a language L , **can you** write a FO formula φ such that $L(\varphi) = L$
 - ▶ If you can, FO is expressive enough to capture your language/specification/property
 - ▶ If you cannot, show that FO cannot capture your property.
- ▶ **Satisfiability**

Expressiveness and Satisfiability

- ▶ Signature for words : $<$, S and Q_a for finitely many symbols a
- ▶ Given a FO formula over words, the signature is fixed
- ▶ **Expressiveness**
 - ▶ Given a set of words or a language L , **can you** write a FO formula φ such that $L(\varphi) = L$
 - ▶ If you can, FO is **expressive enough** to capture your language/specification/property
 - ▶ If you cannot, show that FO cannot capture your property.
- ▶ **Satisfiability**
 - ▶ Given a FO formula φ over words, is $L(\varphi)$ non-empty?
Satisfiable if it is non-empty. If FO formula captures some words then it is satisfiable.

A Primer for Words

Primer: Introductory explanation or fundamental concept.

Alphabet

- ▶ An alphabet Σ is a finite set
 - ▶ $\Sigma = \{a, b, \dots, z\}$
 - ▶ $\Sigma = \{+, \alpha, 100, B\}$
- ▶ Elements of Σ called letters or symbols
- ▶ A word or string over Σ is a finite sequence of symbols from Σ
- ▶ If $\Sigma = \{a, b\}$, then *abababa* is a word of length 7
- ▶ The length of a word w is denoted $|w|$
- ▶ There is a unique word of length 0 denoted ϵ , called the empty word
- ▶ $|\epsilon| = 0$

Notations for Words

- ▶ $aaaaa$ denoted a^5
- ▶ $a^0 = \epsilon$
- ▶ $a^{n+1} = a^n.a = a.a^n$
- ▶ The set of all words over Σ is denoted Σ^*
 - ▶ $\{a, b\}^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, \dots\}$
 - ▶ $\{a\}^* = \{\epsilon, a, aa, aaa, \dots\} = \{a^n \mid n \geq 0\}$
- ▶ By convention, $\{\}^* = \{\epsilon\}$

Notations for Words

- ▶ Σ is a finite set
- ▶ Σ^* is the infinite set of all finite words over alphabet Σ
- ▶ Each $w \in \Sigma^*$ is a finite word
 - ▶ $\{a, b\} = \{b, a\}$ but $ab \neq ba$
 - ▶ $\{a, a, b\} = \{a, b\}$ but $aab \neq ab$
 - ▶ \emptyset is the set consisting of no words
 - ▶ $\{\epsilon\}$ is a set having the single word ϵ
 - ▶ ϵ is a word

Operations on Words

- ▶ Concatenation of words : $x.y = xy$
 - ▶ Concatenation is associative : $x.(yz) = (xy).z$
 - ▶ Concatenation not commutative in general $x.y \neq y.x$
 - ▶ ϵ is the identity for concatenation $\epsilon.x = x.\epsilon = x$
 - ▶ $|x.y| = |x| + |y|$
- ▶ x^n : catenating word x n times
 - ▶ $(aab)^5 = \text{aab aab aab aab aab}$
 - ▶ $(aab)^0 = \epsilon$
 - ▶ $(aab)^* = \{\epsilon, aab, aabaab, aabaabaab, \dots\}$
 - ▶ $x^{n+1} = x^n x$

"*" represents the infinite set.

More Operations on Words

- ▶ For $a \in \Sigma$ and $x \in \Sigma^*$,

$|x|_a =$ *number of times the symbol a occurs in the word x*

- ▶ $|aabbaa|_a = 4, |aabbaa|_b = 2$
- ▶ $|\epsilon|_a = 0$
- ▶ Prefix of a word $w \in \Sigma^*$ is an initial subword of w

$Pref(w) = \{x \in \Sigma^* \mid \exists y \in \Sigma^*, w = x.y\}$
Take words from set which are prefixes.

- ▶ $Pref(aaba) = \{\epsilon, a, aa, aab, aaba\}$
- ▶ Proper prefixes = $\{a, aa, aab\}$
- ▶ $\epsilon, aaba$ improper prefixes

Operation on Sets

Given a finite alphabet Σ , denote by A, B, C, \dots subsets of Σ^*

- ▶ Subsets of Σ^* are called languages
- ▶ $A \cup B = \{x \in \Sigma^* \mid x \in A \text{ or } x \in B\}$
 - ▶ $A = a^*, B = \{b, bb\}, A \cup B = a^* \cup \{b, bb\}$
- ▶ $A \cap B = \{x \in \Sigma^* \mid x \in A \text{ and } x \in B\}$
 - ▶ $A = (ab)^*, B = \{abab, \epsilon, bb\}, A \cap B = \{\epsilon, abab\}$
- ▶ $\bar{A} = \{x \in \Sigma^* \mid x \notin A\}$
 - ▶ For $\Sigma = \{a\}$ and $A = (aa)^*, \bar{A} = \{a, a^3, a^5, \dots\}$ See the empty word is not here.
- ▶ $AB = \{xy \mid x \in A, y \in B\}$
 - ▶ $A = \{a, ba\}, B = \{\epsilon, aa, bb\}$
 - ▶ $AB = \{a, a^3, abb, ba, ba^3, babb\}$
 - ▶ $BA = \{a, ba, a^3, aaba, bba, bbba\}$

Operation on Sets

For a set $A \subseteq \Sigma^*$,

- ▶ $A^0 = \{\epsilon\}$
- ▶ $A^{n+1} = A.A^n$
 - ▶ $\{a, ab\}^2 = \{a, ab\}.\{a, ab\} = \{aa, aab, aba, abab\}$
 - ▶ $\{a, b\}^n = \{x \in \{a, b\}^* \mid |x| = n\}$
- ▶ $A^* = A^0 \cup A \cup A^2 \cup \dots = \bigcup_{i \geq 0} A^i$
- ▶ $A^+ = AA^* = A \cup A^2 \cup \dots = \bigcup_{i \geq 1} A^i$
- ▶ Union : Associative, commutative
- ▶ Concatenation : Associative, Non commutative
- ▶ $A \cup \emptyset = \emptyset \cup A = A$
- ▶ $\{\epsilon\}A = A\{\epsilon\} = A$
- ▶ $\emptyset A = A\emptyset = \emptyset$ Shouldn't it be A .

Operation on Sets

- ▶ Union, Intersection distribute over union

- ▶ $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

- ▶ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

- ▶ Concatenation distributes over union

- ▶ $A(\cup_{i \in I} B_i) = \cup_{i \in I} AB_i$

- ▶ $(\cup_{i \in I} B_i)A = \cup_{i \in I} B_iA$

- ▶ Concatenation does not distribute over intersection

- ▶ $A = \{a, ab\}, B = \{b\}, C = \{\epsilon\}$

- ▶ $A(B \cap C) \neq AB \cap AC$

FO for Languages

Formalize in FO

Write FO formulae φ_i such that $L(\varphi_i) = L_i$ for $i = 1, \dots, 5$.

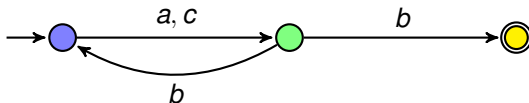
- ▶ L_1 = Words that have exactly one occurrence of the letter c
- ▶ L_2 = Words that begin with a and end with b
- ▶ L_3 = Words that have no two consecutive a 's
- ▶ L_4 = Words in which any a is followed immediately by a b
- ▶ L_5 = Words in which whenever an a occurs, it is followed eventually by a b , and no c occurs in between the a and the b
 $aabbabab, aabbcbbccaab \in L_5, aacaab \notin L_5$.

Satisfiability of FO over Words

- ▶ Recall : Given an FO sentence φ over words, is $L(\varphi) = \emptyset$?
- ▶ Algorithm?

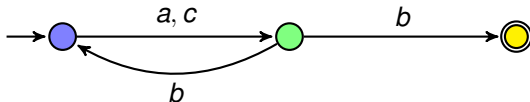
Core Idea

- ▶ Given FO formula φ over an alphabet Σ , construct an **edge labeled graph** G_φ : a graph whose edges are **labeled** by Σ .



Core Idea

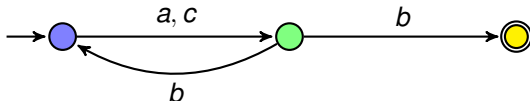
- ▶ Given FO formula φ over an alphabet Σ , construct an **edge labeled graph** G_φ : a graph whose edges are **labeled** by Σ .



- ▶ Each path in the graph gives rise to a word over Σ , obtained by reading off the labels on the edges

Core Idea

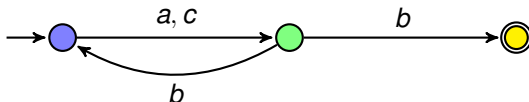
- ▶ Given FO formula φ over an alphabet Σ , construct an **edge labeled graph** G_φ : a graph whose edges are **labeled** by Σ .



- ▶ Each path in the graph gives rise to a word over Σ , obtained by reading off the labels on the edges
- ▶ G_φ has some **special** kinds of vertices
 - ▶ There is a unique vertex called the **start** vertex (blue vertex)
 - ▶ There are some vertices called **good** vertices (yellow vertex)

Core Idea

- ▶ Given FO formula φ over an alphabet Σ , construct an **edge labeled graph** G_φ : a graph whose edges are **labeled** by Σ .



- ▶ Each path in the graph gives rise to a word over Σ , obtained by reading off the labels on the edges
- ▶ G_φ has some **special** kinds of vertices
 - ▶ There is a unique vertex called the **start** vertex (blue vertex)
 - ▶ There are some vertices called **good** vertices (yellow vertex)
- ▶ Read off words on paths from the start vertex to any final vertex and call this set of words $L(G_\varphi)$
- ▶ Ensure that G_φ is constructed such that $L(\varphi) = L(G_\varphi)$.

Core Idea

- ▶ Why does this help?

Core Idea

- ▶ Why does this help?
- ▶ We know how to check the existence of a path between 2 vertices in a graph **easily**

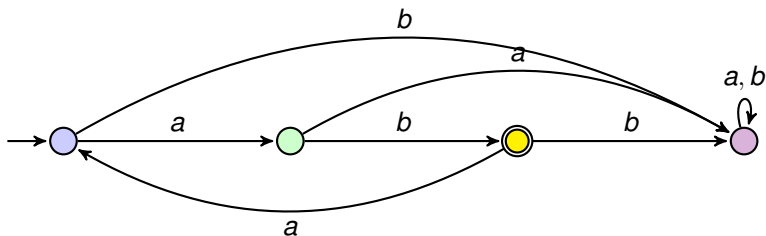
Core Idea

- ▶ Why does this help?
- ▶ We know how to check the existence of a path between 2 vertices in a graph **easily**
- ▶ If **somehow** we manage to construct G_φ **correctly**, then **checking satisfiability of φ is same** as checking the **reachability** of **some good vertex from the start vertex** of G_φ .

Core Idea

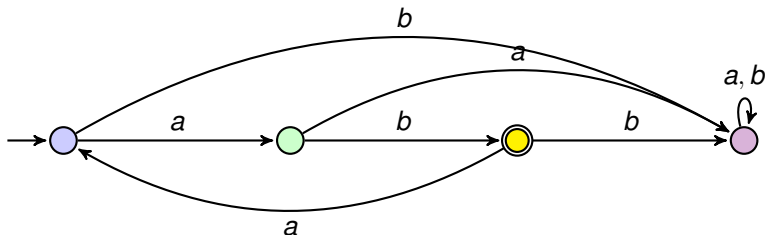
- ▶ Why does this help?
- ▶ We know how to check the existence of a path between 2 vertices in a graph **easily**
- ▶ If **somehow** we manage to construct G_φ **correctly**, then checking satisfiability of φ is same as checking the **reachability** of some good vertex from the start vertex of G_φ .
- ▶ How to construct G_φ ?

A First Machine A



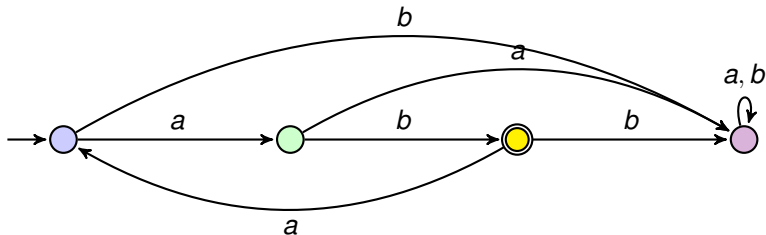
- ▶ Colored circles called **states**
- ▶ Arrows between circles called **transitions**
- ▶ Blue state called an **initial state**
- ▶ Doubly circled state called a **final state**

A First Machine A



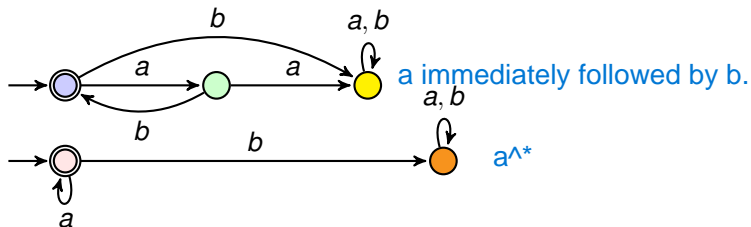
- ▶ A path from one state to another gives a word over $\Sigma = \{a, b, c\}$
- ▶ The machine **accepts** words along paths from an initial state to a final state
- ▶ The set of words accepted by the machine is called the **language** accepted by the machine

A First Machine A



- ▶ What is the language L accepted by this machine, $L(A)$?
- ▶ Write an FO formula φ such that $L(\varphi) = L(A)$

A Second and a Third Machine B, C



- ▶ What are $L(B)$, $L(C)$?
- ▶ Give an FO formula φ such that $L(\varphi) = L(B) \cup L(C)$