A Workbench for Logically Definable Stringsets

Dakotah Lambert Margaret Fero
Andrew Dai James Rogers*

Earlham College

*Faculty Advisor

1 Introduction

2 Types of Logic
- First Order
- Propositional

3 Semantics of Logics
- Satisfaction
- Other Issues

Computational Workbench

Definition

A workbench is a suite of (computational) tools and an interactive environment for using them.

Our workbench includes:

- Tools for defining sets of strings (Formal Languages) with logical constraints
- Tools for reasoning about those logical constraints

First-Order Logical Constraints on Strings

Definition (First-Order Logic)

A first-order logical formula is dependent on the values of variables.

First-Order Logical Constraints on Strings

Definition (First-Order Logic)

A first-order logical formula is dependent on the values of variables.

Example

'a' occurs at some position in the string:

$$(\exists x)\,[a\,(x)]$$

First-Order Logical Constraints on Strings

Definition (First-Order Logic)

A first-order logical formula is dependent on the values of variables.

Example

The substring 'aa' does not occur in the string:

$$(\forall x, y)\,[(a\,(x) \wedge y = x + 1) \implies \neg a\,(y)]$$

First-Order Logical Constraints on Strings

Definition (First-Order Logic)

A first-order logical formula is dependent on the values of variables.

Example

No ‘b’ or ‘c’ occurs following an ‘a’:

$$(\forall x, y)\left[\left(a(x) \wedge x \triangleleft^{+} y\right) \implies \neg\left(b(y) \vee c(y)\right)\right]$$

Propositional Logical Constraints on Strings

Definition (Propositional Logic)

Propositional constraints are independent of position.

Two types:

- Local
- Piecewise

Propositional Logical Constraints on Strings

Definition (Propositional Logic)

Propositional constraints are independent of position.

Two types:

- Local – reasoning with successor only
- Piecewise

Propositional Logical Constraints on Strings

Definition (Propositional Logic)

Propositional constraints are independent of position.

Two types:

- Local – reasoning with successor only
- Piecewise – reasoning with less-than only

Local Propositional Constraints

Local constraints apply to *substrings*, sequences of consecutive symbols. These constraints can include end-markers.

Local Propositional Constraints

Local constraints apply to *substrings*, sequences of consecutive symbols. These constraints can include end-markers.

Example

'aa' does not occur as a substring:

$$\neg aa$$

Local Propositional Constraints

Local constraints apply to *substrings*, sequences of consecutive symbols. These constraints can include end-markers.

Example

The string starts with ‘a’ and ends with ‘b’:

$$\rtimes a \wedge b \ltimes$$

Piecewise Propositional Constraints

Piecewise constraints apply to *subsequences*, sequences of symbols that may not necessarily be consecutive. These constraints cannot include end-markers.

Piecewise Propositional Constraints

Piecewise constraints apply to *subsequences*, sequences of symbols that may not necessarily be consecutive. These constraints cannot include end-markers.

Example

'aa' occurs as a subsequence:

$$a \dots a$$

Piecewise Propositional Constraints

Piecewise constraints apply to *subsequences*, sequences of symbols that may not necessarily be consecutive. These constraints cannot include end-markers.

Example

No ‘b’ or ‘c’ occurs following an ‘a’:

$$\neg\,(a\,..\,b \vee a\,..\,c)$$

Semantics of Logics – Satisfaction ($\models$)

Satisfaction defines a relation between well-formed formulae and models. A model of a formula is the set of strings satisfying that formula, a Formal Language.

Syntax	Models
$(\exists x)\,[a\,(x)]$	Strings containing 'a'
ab	Strings containing the substring 'ab'
$a\,..\,b$	Strings containing the subsequence 'ab'

Systems of Logical Constraints (Axioms)

Definition

A system of logical constraints is a set of logical formulae.

- A model satisfies the system if and only if it satisfies each constraint individually
- The set of models of the system is the intersection of the sets of models of the individual constraints

Semantic Issues

- Consistency:
  - Does a model exist that satisfies the system?
  - If so, is the stringset it describes non-empty?

Semantic Issues

- Consistency:
  - Does a model exist that satisfies the system?
  - If so, is the stringset it describes non-empty?
- Consequece:
  - Is a constraint logically implied by another set of constraints?
  - Does every model that satisfies the set satisfy the constraint?
  - Is the negation of the constraint inconsistent with the set?

Semantic Issues

- Consecuency:
  - Does a model exist that satisfies the system?
  - If so, is the stringset it describes non-empty?
- Consequece:
  - Is a constraint logically implied by another set of constraints?
  - Does every model that satisfies the set satisfy the constraint?
  - Is the negation of the constraint inconsistent with the set?
- Independence
  - Are all of the constraints necessary?
  - Are one or more constraints logical consequences of the others?

Why Formal Logic?

- Transparency

Why Formal Logic?

- Transparency – Logic is used to formalize human reasoning

Why Formal Logic?

- Transparency – Logic is used to formalize human reasoning
- Modularity

Why Formal Logic?

- Transparency – Logic is used to formalize human reasoning
- Modularity
  - Stringsets can be defined with independent constraints
  - We can identify common constraints and constraints that distinguish stringsets

Why Formal Logic?

- Transparency – Logic is used to formalize human reasoning
- Modularity
  - Stringsets can be defined with independent constraints
  - We can identify common constraints and constraints that distinguish stringsets
- Cognitive Complexity

Why Formal Logic?

- Transparency – Logic is used to formalize human reasoning
- Modularity
  - Stringsets can be defined with independent constraints
  - We can identify common constraints and constraints that distinguish stringsets
- Cognitive Complexity
  - The syntactic form characterizes the important properties
  - This determines what a cognitive mechanism (human, animal, or mechanical) must be able to detect in order to distinguish strings that satisfy the system from those that do not

Metrical Stress

pho·´nol·o·gy ` pho·no·´log·i·cal

- Stress Patterns
  - Constraints on the distribution of primary/secondary stress in spoken words
  - Each (stressed) language has a characteristic stress pattern
  - Languages often share the same stress pattern
  - Types of constraints are common across all languages
- Syllable Weight
  - Stress patterns depend only on syllable weight (Light, Heavy, Superheavy,. . .)—abstract categories of syllables.
  - Each language has its own rules for categorizing

An Example: Khmer (Cambodian)

- Primary stress falls on the final syllable
- Secondary stress falls on all heavy syllables
- Light syllables occur only immediately following heavy syllables.
- Light monosyllables do not occur.

Modeling Stress Patterns

Syllables: Alphabet

Weight	Stress				
	None	Primary	Secondary	Some	Any
Any	σ	$\acute{\sigma}$	$\grave{\sigma}$	$\overset{+}{\sigma}$	$\overset{*}{\sigma}$
Light	L	$\acute{L}$	$\grave{L}$	$\overset{+}{L}$	$\overset{*}{L}$
Heavy	H	$\acute{H}$	$\grave{H}$	$\overset{+}{H}$	$\overset{*}{H}$
Super	S	$\acute{S}$	$\grave{S}$	$\overset{+}{S}$	$\overset{*}{S}$
Start: $\rtimes$			End: $\ltimes$		

Words: strings
Constraints: Sets of strings
Stress Patterns: Sets of constraints

Logical Constraints to Automata

Local Atomic Formulae

$$\sigma_1\sigma_2$$

$$\texttt{contains}(\sigma_1 <> \sigma_2)$$

Logical Constraints to Automata

Piecewise Atomic Formulae

$$\sigma_1 \mathrel{..} \sigma_2$$

$$\texttt{contains}(\sigma_1 <> (\texttt{star xss}) <> \sigma_2)$$

Logical Constraints to Automata

Boolean Operations

φ_1 and φ_2 are logical formulae

$\neg\varphi_1$	:	$\texttt{complement}(\varphi_1)$	
$\varphi_1 \wedge \varphi_2$	:	$(\varphi_1 \texttt{ /\textbackslash\textbackslash } \varphi_2)$	intersection
$\varphi_1 \vee \varphi_2$	:	$(\varphi_1 \texttt{ \textbackslash\textbackslash/ } \varphi_2)$	union

Semantic Issues

$$\begin{aligned}\text{satisfaction} &\equiv \text{acceptance}\\ \text{consistency} &\equiv \text{non-emptiness of intersection}\\ \text{consequence} &\equiv \text{inconsistency of } \Phi \wedge \neg\varphi\\ \text{independence} &\equiv \text{no } \varphi \in \Phi \text{ is a consequence of } \Phi - \varphi.\end{aligned}$$

An Example: Khmer

- Primary on final
- Secondary on H
- L only immediately after H
- No $\rtimes L \ltimes$

Primary on Final

$\acute{\sigma} \ltimes \equiv (\neg\, \sigma \ltimes) \wedge (\neg\, \grave{\sigma} \ltimes)$

Also, $(\neg\, \acute{\sigma}\, \overset{*}{\sigma})$:

Secondary on H

(Really just some stress on H)

$$\neg H$$

L only immediately after H

$\neg\, L\, L$ and $\neg \rtimes L$

Implies no light monosyllables.

Khmer as a Whole

The conjunction of everything

Testing Consequence

$$\acute{\sigma}$$

Sub-Regular Classes of Stringsets

- Restricted Propositional
  - Strictly Locally Testable ($\mathbf{SL}_k$, $\mathbf{SL}$)/Strictly Piecewise Testable ($\mathbf{SP}_k$, $\mathbf{SP}$)
  - Conjunctions of forbidden length-k substrings/subsequences

$$\neg abb \wedge \neg b \ltimes \wedge \cdots \qquad \neg(a\,..\,b) \wedge \neg(b\,..\,c\,..\,d) \wedge \cdots$$

  - Only need to be able to sensitive to presence of individual length-k substrings/subsequences in isolation

$$\mathbf{SL} = \bigcup_{0<k} [\mathbf{SL}_k] \qquad \mathbf{SP} = \bigcup_{0<k} [\mathbf{SP}_k]$$

Sub-Regular Classes of Stringsets

- Full Propositional
- Locally Testable ($\mathbf{LT}_k$, $\mathbf{LT}$)/Piecewise Testable ($\mathbf{PT}_k$, $\mathbf{PT}$)
- Boolean combinations of length-k substrings/subsequences

$$(abb \rightarrow \neg b\ltimes) \vee \cdots \qquad (a \mathrel{..} b) \vee \neg(b \mathrel{..} c \mathrel{..} d) \wedge \cdots$$

- Need to be sensitive to the entire set of length-k substrings/subsequences in isolation

$$\mathbf{LT} = \bigcup_{0<k}[\mathbf{LT}_k] \qquad \mathbf{PT} = \bigcup_{0<k}[\mathbf{PT}_k]$$

Other Sub-Regular Classes of Stringsets

- First-Order Logic with successor ($+1$) but not less-than ($<$)
  - Locally (k, t)-Threshold Testable ($\mathbf{LTT}_{k,t}$)
  - Need to be able to count length-k substrings/subsequences up to a fixed threshold t
- First-Order Logic with less-than ($<$)
  - Star-Free Stringsets ($\mathbf{SF}$)
  - Regular expressions without Kleene-$*$ but with complement.
  - Need to be able to detect concatenations of $\mathbf{LT}$ stringsets
- Monadic Second-Order Logic
  - Regular Stringsets ($\mathbf{Reg}$)
  - Need to be able to categorize symbols into finitely many abstract categories depending on category of previous symbol.

Conclusions

- This workbench can provide a viable approach to reasoning about logical constraints
- This directly translates to reasoning about languages, both formal and natural

Future Work

- Full support for first order and monadic second-order logic
- A complete list of currently-known constraints
- A better user-interface