Computational Phonology - Class 7

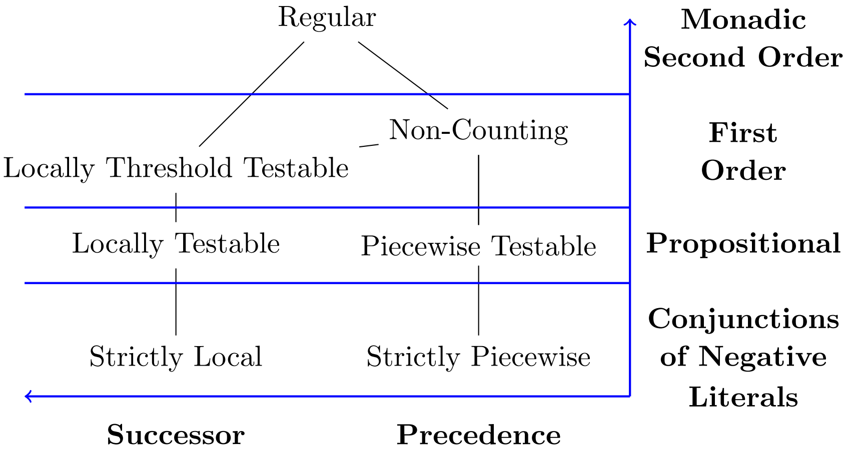
Jeffrey Heinz (Instructor)
Jon Rawski (TA)

LSA Summer Institute
UC Davis
July 15, 2019

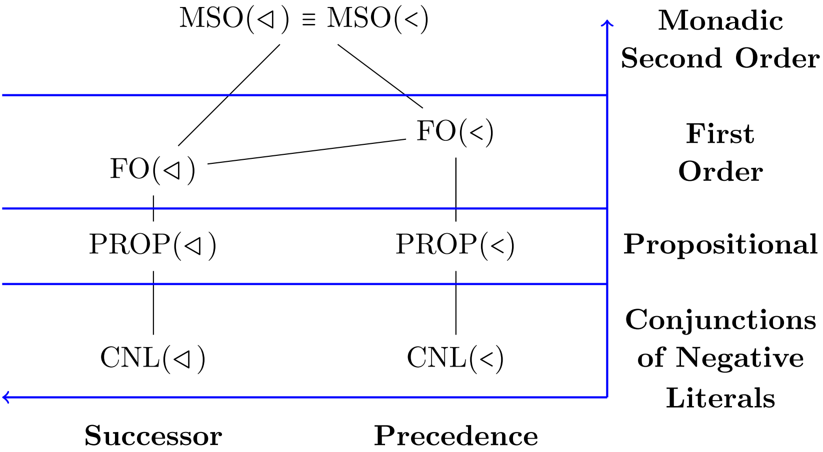
Part I

Review

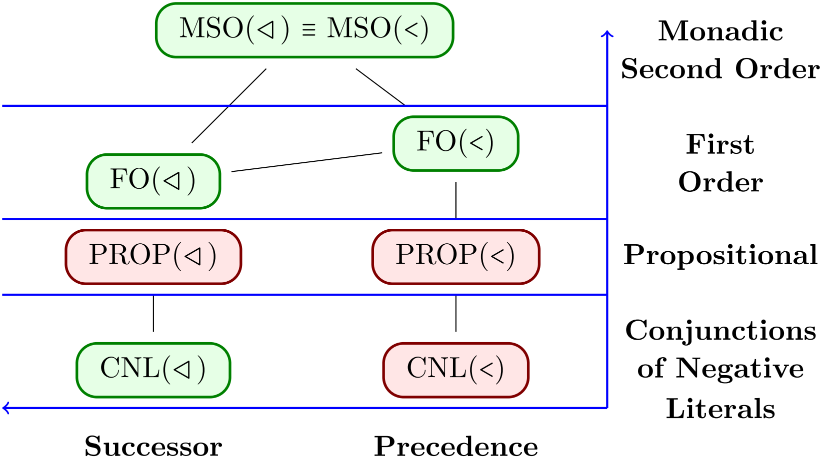
Hierarchies of Constraints



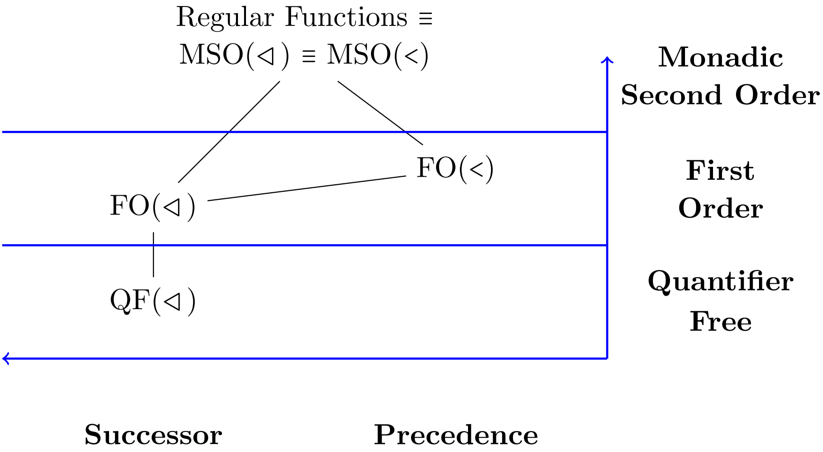
Hierarchies of Constraints



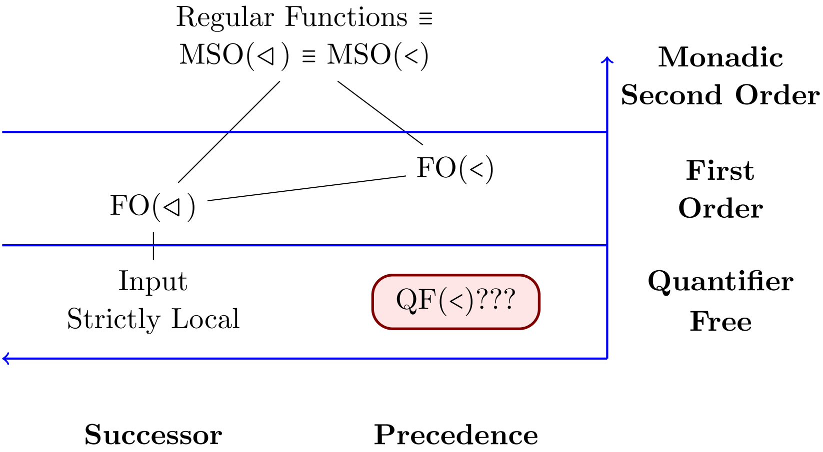
Hierarchies of Constraints



Hierarchies of Transformations



Hierarchies of Transformations



Today

1. Filling out Hierarchies of Constraints
   - PROP($\lhd$)
   - PROP($<$)
   - CNL($<$)
   - ...more representations, more logics?

Today

1. Filling out Hierarchies of Constraints
   - $\boxed{\text{PROP}(\triangleleft)}$
   - $\boxed{\text{PROP}(<)}$
   - $\boxed{\text{CNL}(<)}$
   - ...more representations, more logics?
2. Filling out Hierarchies of Transformations
   - What's the problem with QF(<) transformations?
   - Characterizing Long-distance transformations

Today

1. Filling out Hierarchies of Constraints
   - $\boxed{\text{PROP}(\triangleleft)}$
   - $\boxed{\text{PROP}(<)}$
   - $\boxed{\text{CNL}(<)}$
   - …more representations, more logics?
2. Filling out Hierarchies of Transformations
   - What's the problem with QF($<$) transformations?
   - Characterizing Long-distance transformations
3. Questions and Review regarding everything in the class

Part II

Hierarchies of Constraints

Propositional Logic

Syntax

1. Base cases: There is a set of atomic propositions $\{P, Q, R, S, \ldots\}$, each element of which is a sentence of propositional logic.
2. Inductive case: If φ, ψ are sentences of propositional logic, then so are:
   - $\neg\varphi$ (negation)
   - $\varphi \wedge \psi$ (conjunction)
   - $\varphi \vee \psi$ (disjunction)
   - $\varphi \Rightarrow \psi$ (implication)
   - $\varphi \Leftrightarrow \psi$ (biconditional)

Propositional Logic

Syntax

1. Base cases: There is a set of atomic propositions $\{P, Q, R, S, \ldots\}$, each element of which is a sentence of propositional logic.
2. Inductive case: If φ, ψ are sentences of propositional logic, then so are:
   - $\neg\varphi$ (negation)
   - $\varphi \wedge \psi$ (conjunction)
   - $\varphi \vee \psi$ (disjunction)
   - $\varphi \Rightarrow \psi$ (implication)
   - $\varphi \Leftrightarrow \psi$ (biconditional)

Semantics

1. The atomic propositions are assigned values typically from a truth table.
2. The inductive cases are determined compositionally in the usual way.

With word models, What are the logical atoms?

- The logical atoms (also called *literals*) are *connected structures* in relational models. We call them *factors*.

With word models, What are the logical atoms?

- The logical atoms (also called *literals*) are *connected structures* in relational models. We call them *factors*.
- If a factor $\mathcal{S}$ is contained within $\mathcal{M}_w$ we write $\mathcal{S} \sqsubseteq \mathcal{M}_w$.

With word models, What are the logical atoms?

- The logical atoms (also called *literals*) are *connected structures* in relational models. We call them *factors*.
- If a factor $\mathcal{S}$ is contained within $\mathcal{M}_w$ we write $\mathcal{S} \sqsubseteq \mathcal{M}_w$.
- If $\mathcal{S}$ has k elements in its domain, we say it is of size k and call it a *k-factor*.

With word models, What are the logical atoms?

- The logical atoms (also called *literals*) are *connected structures* in relational models. We call them *factors*.
- If a factor $\mathcal{S}$ is contained within $\mathcal{M}_w$ we write $\mathcal{S} \sqsubseteq \mathcal{M}_w$.
- If $\mathcal{S}$ has k elements in its domain, we say it is of size k and call it a *k-factor*.

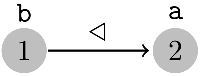
Interpretation of atoms

$\mathcal{M}_w \vDash \mathcal{S}$ if and only if $\mathcal{S} \sqsubseteq \mathcal{M}_w$.

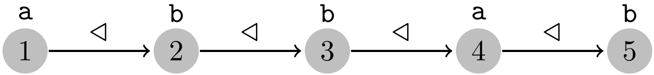
(Rogers and Lambert 2019, Chandlee et al 2019)

Examples

In the successor model, the model of **ba**

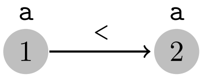


is a factor of the model of **abbab**

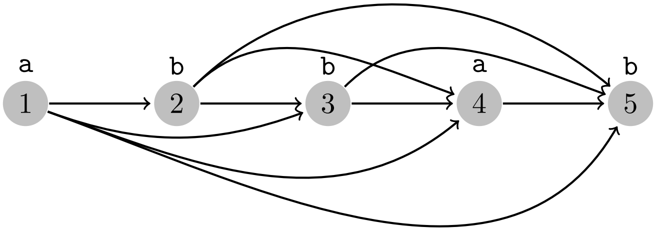


Examples

In the precedence model, the model of **aa**



is a factor of the model of **abbab**



Factors of $\mathcal{M}^{\triangleleft}$ and $\mathcal{M}^{<}$

1. Factors of the *successor* model are essentially *substrings*.

FACTORS OF $\mathcal{M}^{\triangleleft}$ AND $\mathcal{M}^{<}$

1. Factors of the *successor* model are essentially *substrings*.
2. Factors of the *precedence* model are essentially *subsequences*.

(Rogers et al. 2013, Rogers and Lambert 2019)

Locally Testable Constraints ≡ PROP($\mathcal{M}^{\triangleleft}$)

Definition: Locally Testable

A constraint is Locally Testable if and only if it can be defined with propositional logic with the successor model where the atoms are factors of the word models.

Examples of Locally Testable Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa}$?

Examples of Locally Testable Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa}$?
2. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?

Examples of Locally Testable Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa}$?
2. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?
3. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \vee \mathcal{M}_{ab}$?

Examples of Locally Testable Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa}$?
2. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?
3. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \vee \mathcal{M}_{ab}$?
4. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \Rightarrow \mathcal{M}_{ab}$?

Examples of Locally Testable Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa}$?
2. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?
3. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \vee \mathcal{M}_{ab}$?
4. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \Rightarrow \mathcal{M}_{ab}$?
5. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \Leftrightarrow \mathcal{M}_{ab}$?

Examples of Locally Testable Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa}$?
2. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?
3. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \vee \mathcal{M}_{ab}$?
4. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \Rightarrow \mathcal{M}_{ab}$?
5. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \Leftrightarrow \mathcal{M}_{ab}$?
6. Which of the above are Strictly Local?

Abstract Characterization of LT

Theorem: Local Testability (v1)

A constraint is Locally Testable if and only if there is a number k such that for all $u, v \in \Sigma^*$, if u and v have the same set of k-substrings then either both u, v obey the constraint or neither does.

(McNaughton and Papert 1971, Rogers and Pullum 2011, Rogers et al. 2013, Rogers and Lambert 2019)

Abstract Characterization of LT

Theorem: Local Testability (v2)

A constraint is Locally Testable if and only if there is a number k such that for all $u, v \in \Sigma^*$, if the successor models of u and v have the same set of k-factors then either both u, v obey the constraint or neither does.

(McNaughton and Papert 1971, Rogers and Pullum 2011, Rogers et al. 2013, Rogers and Lambert 2019)

Abstract Characterization of LT

Theorem: Local Testability (v2)

A constraint is Locally Testable if and only if there is a number k such that for all $u, v \in \Sigma^*$, if the successor models of u and v have the same set of k-factors then either both u, v obey the constraint or neither does.

1. Provides a way to decide what is NOT Locally Testable.
2. Provides inference rules for learning.
3. Provides a characterization independent of any formalism.

(McNaughton and Papert 1971, Rogers and Pullum 2011, Rogers et al. 2013, Rogers and Lambert 2019)

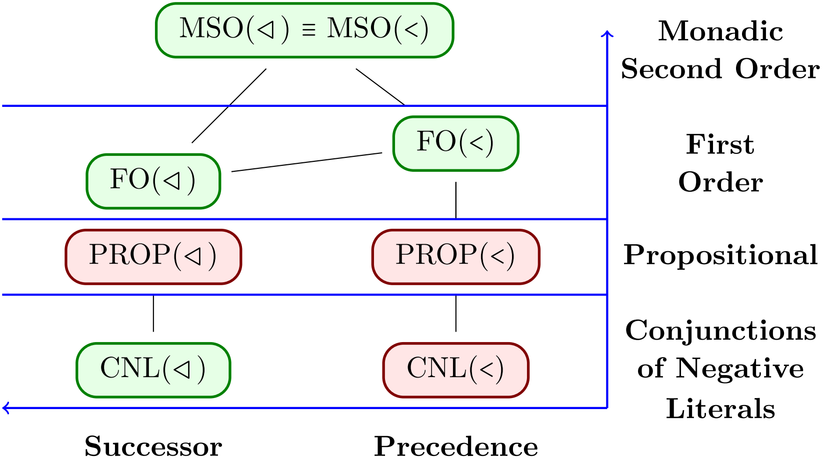
In class Exercises

1. Prove that the constraint which says “Words have at least two *a*s” is not LT.

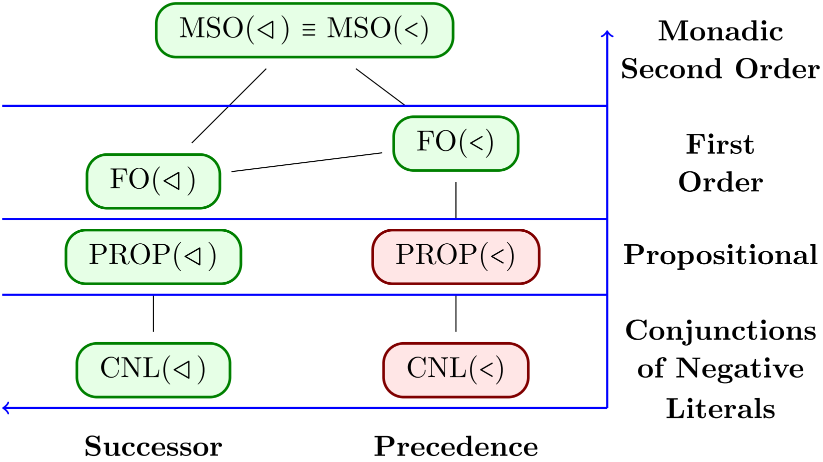
In class Exercises

1. Prove that the constraint which says "Words have at least two *a*s" is not LT.
2. You are born on planet Locally 2-Testable, where everyone's DNA has a UG programmed for LT-2 constraints. You observe the word *aab*.
   1. Can you infer that *aaab* is a word in your language?
   2. What about *ab*?
   3. Do inhabitants of planet Strictly 2-Local generalize in the same way?

Hierarchies of Constraints



Hierarchies of Constraints



Piecewise Testable Constraints ≡ PROP($\mathcal{M}^<$)

Definition: Piecewise Testable

A constraint is Piecewise Testable if and only if it can be defined with propositional logic with the precedence model where the atoms are factors of the word models.

Examples of Piecewise Testable Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa}$?

Examples of Piecewise Testable Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa}$?
2. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?

Examples of Piecewise Testable Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa}$?
2. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?
3. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \vee \mathcal{M}_{ab}$?

Examples of Piecewise Testable Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa}$?
2. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?
3. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \vee \mathcal{M}_{ab}$?
4. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \Rightarrow \mathcal{M}_{ab}$?

Examples of Piecewise Testable Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa}$?
2. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?
3. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \vee \mathcal{M}_{ab}$?
4. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \Rightarrow \mathcal{M}_{ab}$?
5. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \Leftrightarrow \mathcal{M}_{ab}$?

Examples of Piecewise Testable Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa}$?
2. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?
3. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \vee \mathcal{M}_{ab}$?
4. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \Rightarrow \mathcal{M}_{ab}$?
5. $\varphi \stackrel{\text{def}}{=} \mathcal{M}_{aa} \Leftrightarrow \mathcal{M}_{ab}$?

Abstract Characterization of PT

Theorem: Piecewise Testability (v1)

A constraint is Locally Testable if and only if there is a number k such that for all $u, v \in \Sigma^*$, if u and v have the same k-subsequences then either both u, v obey the constraint or neither does.

(Simon 1975, Rogers et al. 2013, Rogers and Lambert 2019)

Abstract Characterization of PT

Theorem: Piecewise Testability (v2)

A constraint is Locally Testable if and only if there is a number k such that for all $u, v \in \Sigma^*$, if the precedence models of u and v have the same k-factors then either both u, v obey the constraint or neither does.

(Simon 1975, Rogers et al. 2013, Rogers and Lambert 2019)

Abstract Characterization of PT

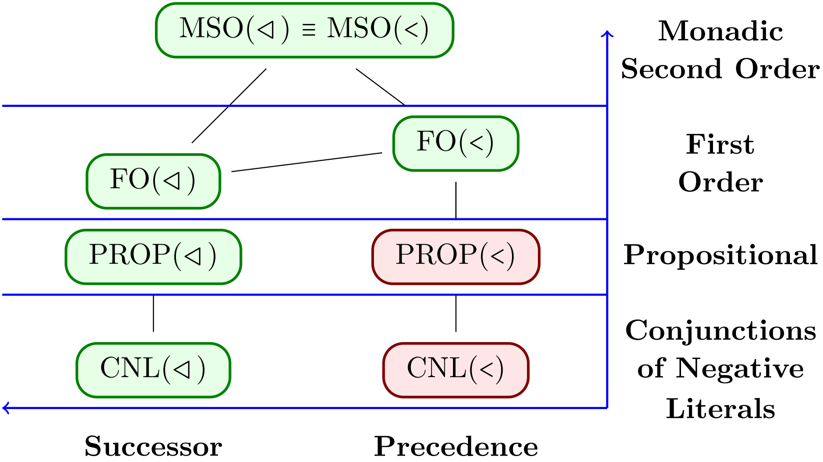
Theorem: Piecewise Testability (v2)

A constraint is Locally Testable if and only if there is a number k such that for all $u, v \in \Sigma^*$, if the precedence models of u and v have the same k-factors then either both u, v obey the constraint or neither does.

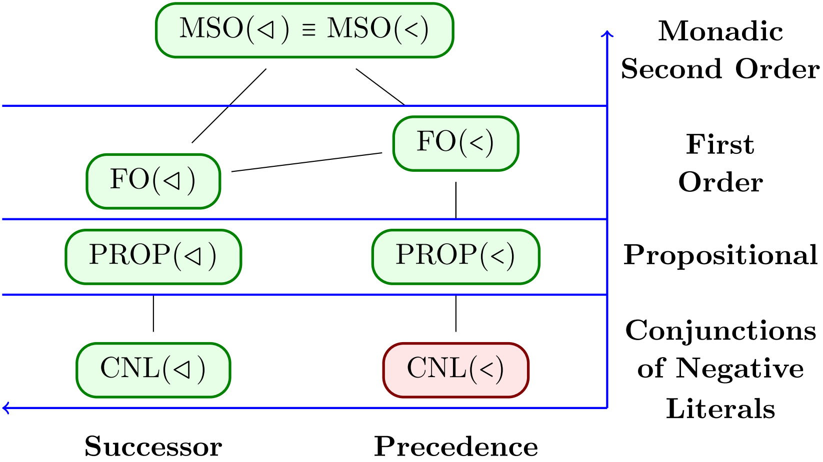
1. Provides a way to decide what is NOT Piecewise Testable.
2. Provides inference rules for learning.
3. Provides a characterization independent of any formalism.

(Simon 1975, Rogers et al. 2013, Rogers and Lambert 2019)

Hierarchies of Constraints



Hierarchies of Constraints



Strictly Piecewise Testable Constraints ≡ CNL($\mathcal{M}^<$)

Definition: Stricty Piecewise

A constraint is Strictly Piecewise if and only if it can be defined as the conjunctions of negative literals within a propositional logic with the precedence model where the atoms are factors of the word models.

Examples of Strictly Piecewise Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?

Examples of Strictly Piecewise Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?
2. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa} \wedge \mathcal{M}_{bb}$?

Examples of Strictly Piecewise Constraints

Which words have models that satisfy φ?

1. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa}$?
2. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa} \wedge \mathcal{M}_{bb}$?
3. $\varphi \stackrel{\text{def}}{=} \neg\mathcal{M}_{aa} \wedge \mathcal{M}_{bb} \wedge \mathcal{M}_{ab}$?

Abstract Characterization of SP

Subsequence Closure

A constraint is Strictly Piecewise if and only if it is closed under subsequence.

In other words, if a string u obeys the constraint then every one of its subsequences also obeys the constraint.

1. Provides a way to decide what is NOT Strictly Piecewise.
2. Provides inference rules for learning.
3. Provides a characterization independent of any formalism.

(Rogers et al. 2010, 2013)

Filling out Hierarchies of Constraints

- $\boxed{\text{PROP}(\triangleleft)}$
- $\boxed{\text{PROP}(<)}$
- $\boxed{\text{CNL}(<)}$
- …more representations, more logics?

Filling out Hierarchies of Constraints

Tier-based Strictly Local

Filling out Hierarchies of Constraints

Tier-based Strictly Local

- This is CNL($\mathcal{M}^{\triangleleft_T}$).
- So a verion of Suffix-Substition Closure applies here.

Filling out Hierarchies of Constraints

Tier-based Strictly Local

- This is CNL($\mathcal{M}^{\triangleleft_T}$).
- So a verion of Suffix-Substition Closure applies here.

Multiple Tier-based Strictly Local

Filling out Hierarchies of Constraints

Tier-based Strictly Local

- This is CNL($\mathcal{M}^{\triangleleft_T}$).
- So a verion of Suffix-Substition Closure applies here.

Multiple Tier-based Strictly Local

- Is this CNL($\mathcal{M}^{\triangleleft_{T_1}, \triangleleft_{T_2}, \ldots \triangleleft_{T_n}}$)?

Filling out Hierarchies of Constraints

Tier-based Strictly Local

- This is CNL($\mathcal{M}^{\triangleleft_T}$).
- So a verion of Suffix-Substition Closure applies here.

Multiple Tier-based Strictly Local

- Is this CNL($\mathcal{M}^{\triangleleft_{T_1},\triangleleft_{T_2},\ldots\triangleleft_{T_n}}$)?

Structure-Sensitive Tier-based Strictly Local

Filling out Hierarchies of Constraints

Tier-based Strictly Local

- This is CNL($\mathcal{M}^{\triangleleft_T}$).
- So a verion of Suffix-Substition Closure applies here.

Multiple Tier-based Strictly Local

- Is this CNL($\mathcal{M}^{\triangleleft_{T_1}, \triangleleft_{T_2}, \dots \triangleleft_{T_n}}$)?

Structure-Sensitive Tier-based Strictly Local

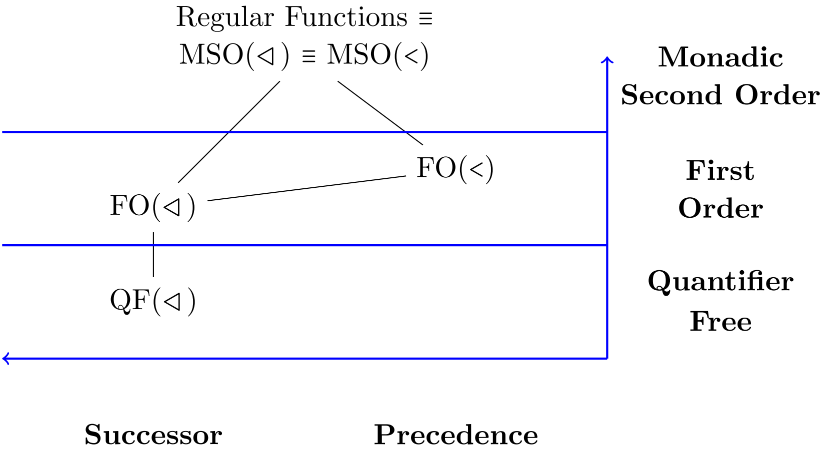
- My guess is this is also CNL with a successor relation that is defined according to the structure-sensitive tier-projection. Somebody ought to look into it!

(Heinz et al. 2011, DeSanto and Graf 2019)

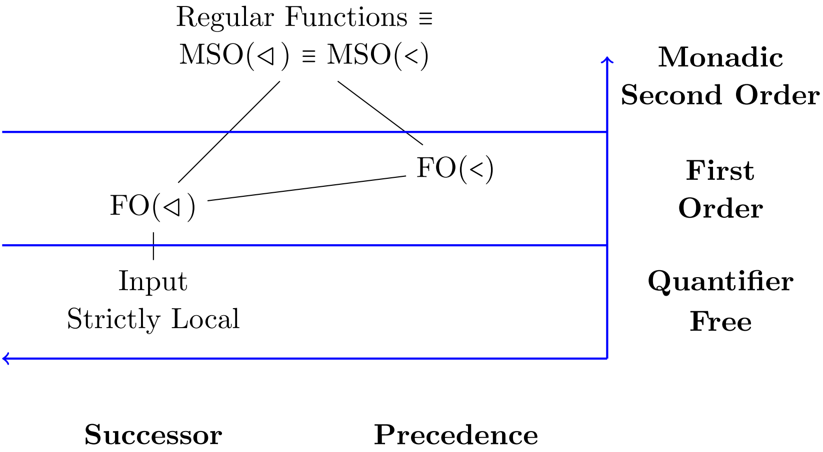
Part III

Hierarchies of Transformations

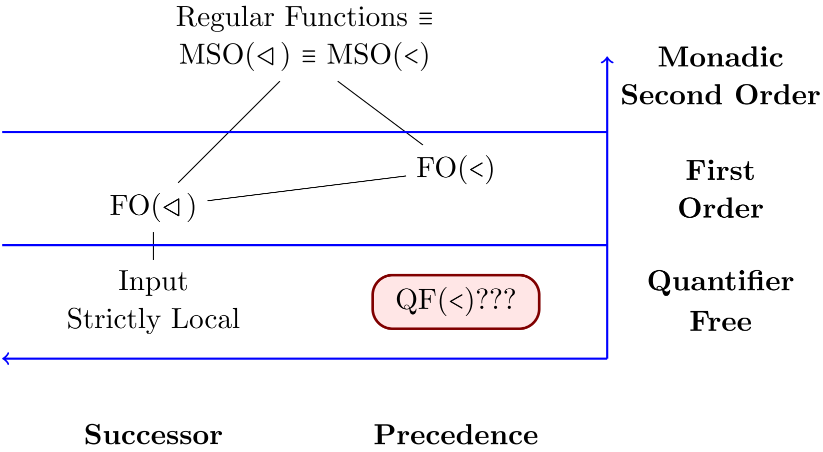
Hierarchies of Transformations



Hierarchies of Transformations



Hierarchies of Transformations



Locality and Quantification

Compare:

1. $P(x) \stackrel{\text{def}}{=} Q(x) \wedge \exists y[R(y)]$ (First Order Definable)

   Requires scanning whole word for such a y!!

2. $P(x) \stackrel{\text{def}}{=} Q(x) \wedge R(\texttt{successor}(x))$ (QF Definable)

   Information to decide P is local to x in the input!!

Logical Characterization of ISL Functions

Theorem.

ISL functions correspond exactly to the functions definable with Quantifier-Free logic with successor: QF($\triangleleft$).

(Lindell and Chandlee, LICS 2016)

Logical Characterization of ISL Functions

Theorem.

ISL functions correspond exactly to the functions definable with Quantifier-Free logic with successor: QF($\triangleleft$).

Quantifier-free formulas are ones without *any* quantification!

(Lindell and Chandlee, LICS 2016)

The successor relation is a function.

1. Each element *has at most one* successor.

The successor relation is a function.

1. Each element *has at most one* successor.

$$\lhd \stackrel{\text{def}}{=} \{(1,2),(2,3),(3,4)\}$$

The successor relation is a function.

1. Each element *has at most one* successor.

$$\triangleleft \stackrel{\text{def}}{=} \{(1,2),(2,3),(3,4)\}$$

2. Therefore, we can use the successor *function* instead of the successor *relation* in the signature of the model.

The successor relation is a function.

1. Each element *has at most one* successor.

$$\triangleleft \stackrel{\text{def}}{=} \{(1,2),(2,3),(3,4)\}$$

2. Therefore, we can use the successor *function* instead of the successor *relation* in the signature of the model.

The payoff

Instead of

$$C_xC \stackrel{\text{def}}{=} \texttt{cons}(x) \wedge \exists y[x \triangleleft\ y \wedge \texttt{cons}(y)]$$

The successor relation is a function.

1. Each element *has at most one* successor.

$$\triangleleft \stackrel{\text{def}}{=} \{(1,2),(2,3),(3,4)\}$$

2. Therefore, we can use the successor *function* instead of the successor *relation* in the signature of the model.

The payoff
Instead of

$$C_xC \stackrel{\text{def}}{=} \texttt{cons}(x) \wedge \exists y[x \triangleleft\ y \wedge \texttt{cons}(y)]$$

We can write

$$C_xC \stackrel{\text{def}}{=} \texttt{cons}(x) \wedge \texttt{cons}(\texttt{succ}(x))]$$

The successor relation is a function.

1. Each element *has at most one* successor.

$$\lhd \stackrel{\text{def}}{=} \{(1,2),(2,3),(3,4)\}$$

2. Therefore, we can use the successor *function* instead of the successor *relation* in the signature of the model.

The payoff

Instead of

$$C_x C \stackrel{\text{def}}{=} \texttt{cons}(x) \wedge \exists y[x \lhd y \wedge \texttt{cons}(y)]$$

We can write

$$C_x C \stackrel{\text{def}}{=} \texttt{cons}(x) \wedge \texttt{cons}(\texttt{succ}(x))]$$

No quantification needed to introduce elements local to x!

The precedence relation is NOT a function.

1. Each element *may precede more than one* element.

The precedence relation is NOT a function.

1. Each element *may precede more than one* element.

$$\lhd \stackrel{\text{def}}{=} \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$$

The precedence relation is NOT a function.

1. Each element *may precede more than one* element.

$$\lhd \stackrel{\text{def}}{=} \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$$

2. Therefore, we cannot use the precedence *function* in the signature of the model – there is literally no such thing!

The precedence relation is NOT a function.

1. Each element *may precede more than one* element.

$$\lhd \overset{\text{def}}{=} \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$$

2. Therefore, we cannot use the precedence *function* in the signature of the model – there is literally no such thing!

Open Questions

- How can the notion of precedence be employed to describe transformations?

The precedence relation is NOT a function.

1. Each element *may precede more than one* element.

$$\lhd \stackrel{\text{def}}{=} \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$$

2. Therefore, we cannot use the precedence *function* in the signature of the model – there is literally no such thing!

Open Questions

- How can the notion of precedence be employed to describe transformations?

1. Tiers: Chandlee and McMullin (2018), Burness and McMullin (2019)
2. Least fixed point logics: Chandlee and Jardine (2019)

The precedence relation is NOT a function.

1. Each element *may precede more than one* element.

$$\triangleleft \stackrel{\text{def}}{=} \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$$

2. Therefore, we cannot use the precedence *function* in the signature of the model – there is literally no such thing!

Open Questions

- How can the notion of precedence be employed to describe transformations?

1. Tiers: Chandlee and McMullin (2018), Burness and McMullin (2019)
2. Least fixed point logics: Chandlee and Jardine (2019)
3. Plenty of room for new ideas here!

Summary

1. Using model signatures with functions instead of relations is a key element to obtaining Quantifier-Free transformations.
2. Characterizing long-distance transformations is a challenging frontier.
3. The constraint hierarchies have two levels below FO: Prop and CNL. The transformation hierarchies only have one level QF, which appears to correspond to CNL. What fragment of FO in the transformation hierarchy corresponds to the Prop level in the constraint hierarchy?

Part IV

Questions regarding anything in class

You fill in the slide here

Summary

Today

1. We completed the basic hierarchies of constraints over strings.

Summary

Today

1. We completed the basic hierarchies of constraints over strings.
2. More to explore!

Summary

Today

1. We completed the basic hierarchies of constraints over strings.
2. More to explore!
3. We reviewed the transformation hierarchies.
4. Much more to do there!

Summary

Today

1. We completed the basic hierarchies of constraints over strings.
2. More to explore!
3. We reviewed the transformation hierarchies.
4. Much more to do there!

Summary

Today

1. We completed the basic hierarchies of constraints over strings.
2. More to explore!
3. We reviewed the transformation hierarchies.
4. Much more to do there!

Next class

1. Computational Theories of Learning?