

2018/12/10 18:20 修正

計算言語学₉

文の意味表現

東京大学生産技術研究所

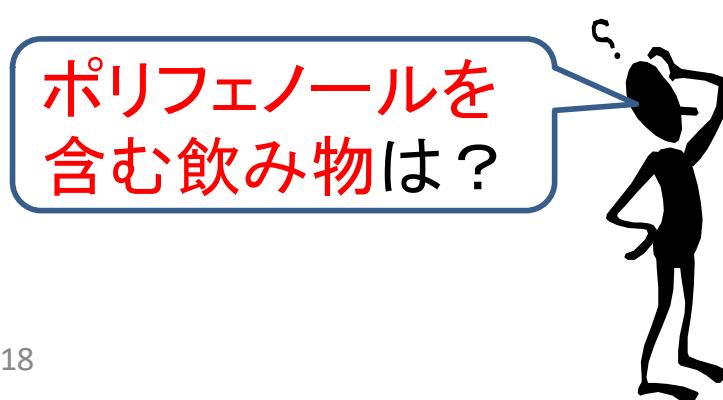
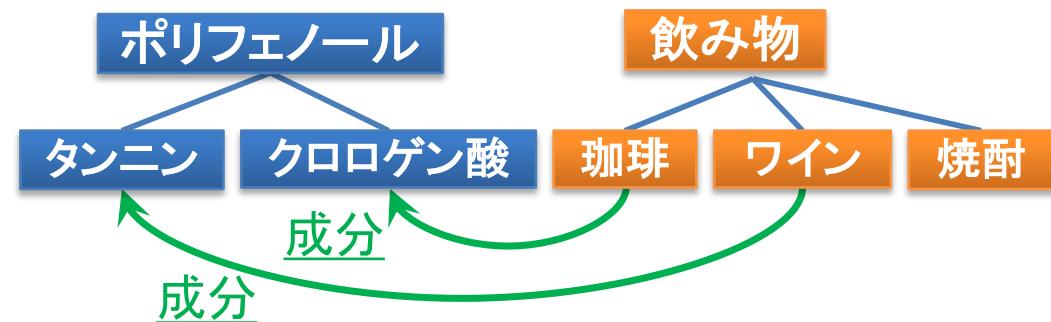
吉永 直樹

site: <http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/class/cl/>

自然言語の意味をどう扱うか？(再考)

- ・言語では膨大な種類の語を組み合わせて文を構成
 - ・Google n-gram での異なり 1-gram 数: 2,565,424
- ・異なる言語表現間の意味的関係をどう捉えるか？

語の意味的な関係



自然言語の意味を扱うための基本方針

- 構成的意味論 (compositional semantics)

The meaning of a complex expression is a function of the meaning of its parts and the way those parts are (syntactically) combined (Freige, 1892)

- 仮定: 文の意味は部分(単語) から構成的に計算可能
- 意味の合成方法は個別の単語の意味とは独立に規定
例) ラムダ計算, 算術平均 など

- 復習: 単語の意味を表現するための2つの方法:

<http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/class/cl/cl4.pdf>

- 意味 = 記号(の組み合わせ): 語彙意味論 (lexical semantics)
- 意味 = 実数値ベクトル: ベクトル意味論 (vector semantics)

Q: 単語の意味から句・文の意味をどう合成するか?

文の記号的意味表現

言語表現の記号的意味表現 (1/2)

- 言語表現の表層的・構文的な違いを捨象した意味に関するデータ構造
- 意味を考慮する様々な自然言語タスクを解く際,
言語表現と実世界の知識の間の橋渡しをする
例)
 - レストランでメニューを読んで何を注文するか決める
 - マニュアルを読んで新しいソフトウェアの使い方を学ぶ
 - 試験で論述問題を解く (cf. 東口ボくん, 高考機器人)
 - 自分が侮辱されたことを理解する
 - レシピに従って料理を作る

言語表現の記号的意味表現 (2/2)

- 意味表現は以下の2つのモデル化を行う
 - 言語表現(句・文)の意味
 - 実世界の状態(世界知識)

注) 意味表現では字句通りの意味を扱い、話者との関係によって決まる語用論的意味は対象外
- 意味解析: 言語表現(句・文) → 意味表現
- 自然言語処理で用いられる記号論的意味表現
 - 一階述語論理(First-Order Logic)
 - 記述論理(Description Logics)
 - Abstract Meaning Representation (AMR) [Banarescu+ 2013]
 - 知識フレーム, Slot-Filler構造

意味表現に何が必要か (1/5): 検証可能性

- 旅行客の質問に、関係するドメインの知識ベースを用いて回答するシステムを考える
 - 知識ベース: 実世界の状態 (世界知識)
- 以下の質問に答えるには何が必要か？

Does Maharani serve vegetarian food?

Serves(Maharani, VegetarianFood)

- 要件1: 意味表現で表現された事態間について、その関係を比較・検証できる必要がある

意味表現に何が必要か (2/5): 非曖昧性

- 以下の文の意味には本質的に曖昧性がある

*I wanna eat **someplace** that's close to Kamata.*

- 蒲田近くのレストランを探している
 - 蒲田近くの土地を壊滅したい（シン・ゴジラ的解釈）
-
- 要件2: 言語表現の意味表現は(最終的には)一意でなければならない
 - ただし、意味の抽象性 (vagueness) はあっても良い

I want to eat Italian food.

意味表現に何が必要か(3/5): 標準形

- 以下の文の意味表現はどうあるべきか？

*Does Maharani **have vegetarian dishes**?*

*Do they **have vegetarian food** at Maharani?*

*Are **vegetarian dishes served** at Maharani?*

*Does Maharani **serve vegetarian fare**?*

- 要件3: (知識ベースの事実との照合のため) 意味が
同じ言語表現は同一の意味表現を持つべき
 - 意味表現は標準形を持つ
 - 標準形への変換には語義曖昧性解消や構文解析が有用

意味表現に何が必要か (4/5): (推論を実現する) 変数

- 以下の質問に答えるには何が必要か？

I'd like to find a restaurant where I can get vegetarian food.

Serves (x, VegetarianFood)

- 要件4: 意味表現は変数に基づく推論を提供する
 - 知識ベースとの照合に基づき変数の値を埋める

意味表現に何が必要か(5/5): 表現力

- 要件5: 意味表現は多様な言語表現の意味の違いを表現し分ける表現力が求められる
- これらの要件を満たす意味表現言語として,
一階述語論理 (First-Order Logic) を以下で見ていく

モデル理論的意味論 (Model-Theoretic Semantics)

- モデル = 実世界の形式化
 - オブジェクトの集合(ドメイン), オブジェクトの性質, オブジェクト間の関係をそれぞれ表現
 - 多くの意味表現言語で仮定
- 意味表現の語彙(構成要素)
 - 非論理的語彙: (字句通りの)明示的意味を指す
オブジェクトの名前と性質, オブジェクト間の関係
 - 論理的語彙:
記号, 演算子, 量化子, リンクなど

モデル理論的意味論: 例 レストラン世界のモデル化

- Domain オブジェクトの集合 $D = \{a,b,c,d,e,f,g,h,i,j\}$
Matthew, Franco, Katie, Caroline
Frasca, Med, Rio
Italian, Mexican, Eclectic
 a, b, c, d
 e, f, g
 h, i, j
- Properties 特定の性質を有するオブジェクトの集合(の集合)
 $Noisy = \{e, f, g\}$
Frasca, Med, and Rio are noisy
- Relations 特定の関係を有するオブジェクトのペアの集合
 $Likes = \{\langle a, f \rangle, \langle c, f \rangle, \langle c, g \rangle, \langle b, e \rangle, \langle d, f \rangle, \langle d, g \rangle\}$
Mathew likes the Med
Katie likes the Med and Rio
Franco likes Frasca
Caroline likes the Med and Rio
- Serves
 $Serves = \{ \langle f, j \rangle, \langle g, i \rangle, \langle e, h \rangle \}$
Med serves eclectic
Rio serves Mexican
Frasca serves Italian

意味表現言語はモデルの元で文
の真偽値を計算する方法を提供

一階述語論理 (First-Order Logic; FOL)

- 意味表現の要件を満たす計算容易な意味表現言語
 - オブジェクトとその性質, オブジェクト間の関係を表現

<i>Formula</i>	\rightarrow	<i>AtomicFormula</i>
		<i>Formula Connective Formula</i>
		<i>Quantifier Variable, … Formula</i>
		\neg <i>Formula</i>
		(<i>Formula</i>)
<i>AtomicFormula</i>	\rightarrow	<i>Predicate(Term, …)</i>
<i>Term</i>	\rightarrow	<i>Function(Term, …)</i>
		<i>Constant</i>
		<i>Variable</i>
<i>Connective</i>	\rightarrow	\wedge \vee \Rightarrow
<i>Quantifier</i>	\rightarrow	\forall \exists
<i>Constant</i>	\rightarrow	<i>A</i> <i>VegetarianFood</i> <i>Maharani</i> …
<i>Variable</i>	\rightarrow	<i>x</i> <i>y</i> …
<i>Predicate</i>	\rightarrow	<i>Serves</i> <i>Near</i> …
<i>Function</i>	\rightarrow	<i>LocationOf</i> <i>CuisineOf</i> …

一階述語論理 (First-Order Logic; FOL)

- **述語 (predicate):** 項を引数に取り状態や動作を指す
Serves(Maharani, VegetarianFood)
- **項 (term):**
 - 定数 (constants): 具体的なオブジェクトを指す記号
Maharani, Henry, etc.
 - 関数 (functions): 所有格を項を取る関数として表現
LocationOf(Frasca) = Frasca's location
 - 変数 (variables): 任意のオブジェクトと照合可能な記号

一階述語論理 (First-Order Logic; FOL): 例

- *Maharani serves vegetarian food*
Serves(Maharani, VegetarianFood)
- *Maharani is a restaurant*
Restaurant(Maharani)
- *I only have five dollars and I don't have a lot of time*
Have(Speaker, FiveDollar) \wedge \neg Have(Speaker, LotOfTime)

一階述語論理 (First-Order Logic; FOL): 変数と量化子

- 変数: 量化子により任意の匿名オブジェクトまたは全オブジェクトを指す

x, y, \dots

- 量化子: \exists (存在量化子), \forall (全称量化子)
a restaurant that serves Mexican food near ICSI

$$\begin{aligned} & \exists x \text{ Restaurant}(x) \\ & \wedge \text{Serves}(x, \text{MexicanFood}) \\ & \wedge \text{Near}(\text{LocationOf}(x), \text{LocationOf}(ICSI)) \end{aligned}$$

All vegetarian restaurants serve vegetarian food

$$\forall x \text{ VegetarianRestaurant}(x) \Rightarrow \text{Serves}(x, \text{VegitarianFood})$$

一階述語論理 (First-Order Logic; FOL): ラムダ記法 [Church 1940]

- 述語における関数適用を表現

$$\lambda x. P(x)$$

$$\lambda x. P(x)(A) \Rightarrow P(A) \quad \underline{\lambda\text{-reduction}}$$

- Currying: 複数の項を取る述語に対して
逐次的な関数適用を実現

$$\lambda x. \lambda y. Near(x, y)$$

$$\lambda x. \lambda y. Near(x, y)(Bacaro) \Rightarrow \lambda y. Near(Bacaro, y)$$

$$\lambda y. Near(Bacaro, y)(Centro) \Rightarrow Near(Bacaro, Centro)$$

一階述語論理に基づく意味論

- モデル理論的アプローチにより、一階述語論理で実世界をモデルとして形式化
- 与えられた言語表現の真偽値を、モデルのもとで以下の論理演算表に基づき計算

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
False	False	True	False	False	True
False	True	True	False	True	True
True	False	False	False	True	False
True	True	False	True	True	True

一階述語論理に基づく推論(演繹)

- Modus ponens: *if-then* 基づく推論

$$\frac{\alpha}{\frac{\alpha \rightarrow \beta}{\beta}}$$

VegetarianRestautrant(Leaf)

$\frac{\forall x \text{ VegetarianRestautant}(x) \rightarrow \text{Serves}(x, \text{VegetarianFood})}{\text{Serves}(\text{Leaf}, \text{VegetarianFood})}$

- Forward chaining (前向き推論): α and $\alpha \Rightarrow \beta$, then β
- Backward chaining (後向き推論): if β in not in KB, then search α
- modus ponens の逆向き適用 (plausible reasoning) 基づく推論を仮説的推論 (abduction) と呼ぶ

一階述語論理に基づく推論(演繹): 例

- $Serves(Leaf, VegetarianFood)$ の真偽値計算

知識ベース (KB)

$\forall x \text{VegetarianRestautant}(x) \rightarrow Serves(x, \text{VegetarianFood})$

$\text{VegetarianRestautrant}(Leaf)$

- $\forall x \text{VegetarianRestautant}(x) \rightarrow Serves(x, \text{VegetarianFood})$ の帰結部に $x = Leaf$ を代入し, 得られた前提が KB にあるかを探索

Neo-Davidsonian 意味論 (1/3): イベントや状態の意味表現

- 持続する状態 (state) や実世界の状態の変化 (event), またそれらの時間的関係を記述したい

After eating, I slept.

- 異なる数の項を取る述語の間の関係を記述したい

I eat.

I eat a turkey sandwich.

I eat a turkey sandwich at my desk.

- Neo-Davidsonian 意味論 [Davidson 1967]
一階述語論理にイベント変数を導入

$\exists e Eating(e)$

Neo-Davidsonian 意味論 (2/3): イベントや状態の意味表現

- 述語が関係するイベントを表現する イベント変数 を導入し、述語の項として扱う

I eat a turkey sandwich at my desk.

Eat(Speaker, TurkeySandwich, Desk)

$$\begin{array}{l} \rightarrow \exists e Eating(e) \\ \quad \wedge Eater(e, Speaker) \\ \quad \wedge Eaten(e, TurkeySandwich) \end{array}$$

- 取る項のパターン(下位範疇化フレーム)により同じ意味の動詞の述語を分ける必要がなくなる

Neo-Davidsonian 意味論 (3/3): 時間の意味表現

- イベント変数により時間を表現する事が可能に

I arrived in New York.

$$\exists e, i, n \text{ Arriving}(e) \wedge \text{Arriver}(e, \text{Speaker}) \wedge \text{Destination}(e, \text{New York})$$
$$\wedge \text{IntervalOf}(e, i) \wedge \text{Endpoint}(i, n) \wedge \text{Proceeds}(n, \text{Now})$$

イベントのインターバル イベントの終了時間

I will arrive in New York.

$$\exists e, i, n \text{ Arriving}(e) \wedge \text{Arriver}(e, \text{Speaker}) \wedge \text{Destination}(e, \text{New York})$$
$$\wedge \text{IntervalOf}(e, i) \wedge \text{Endpoint}(i, n) \wedge \text{Proceeds}(\text{Now}, n)$$

- 現在完了など複雑な時制についても参照ポイント [Reichenbach 1947] を考えることで表現可能 (省略)

アスペクトの意味表現

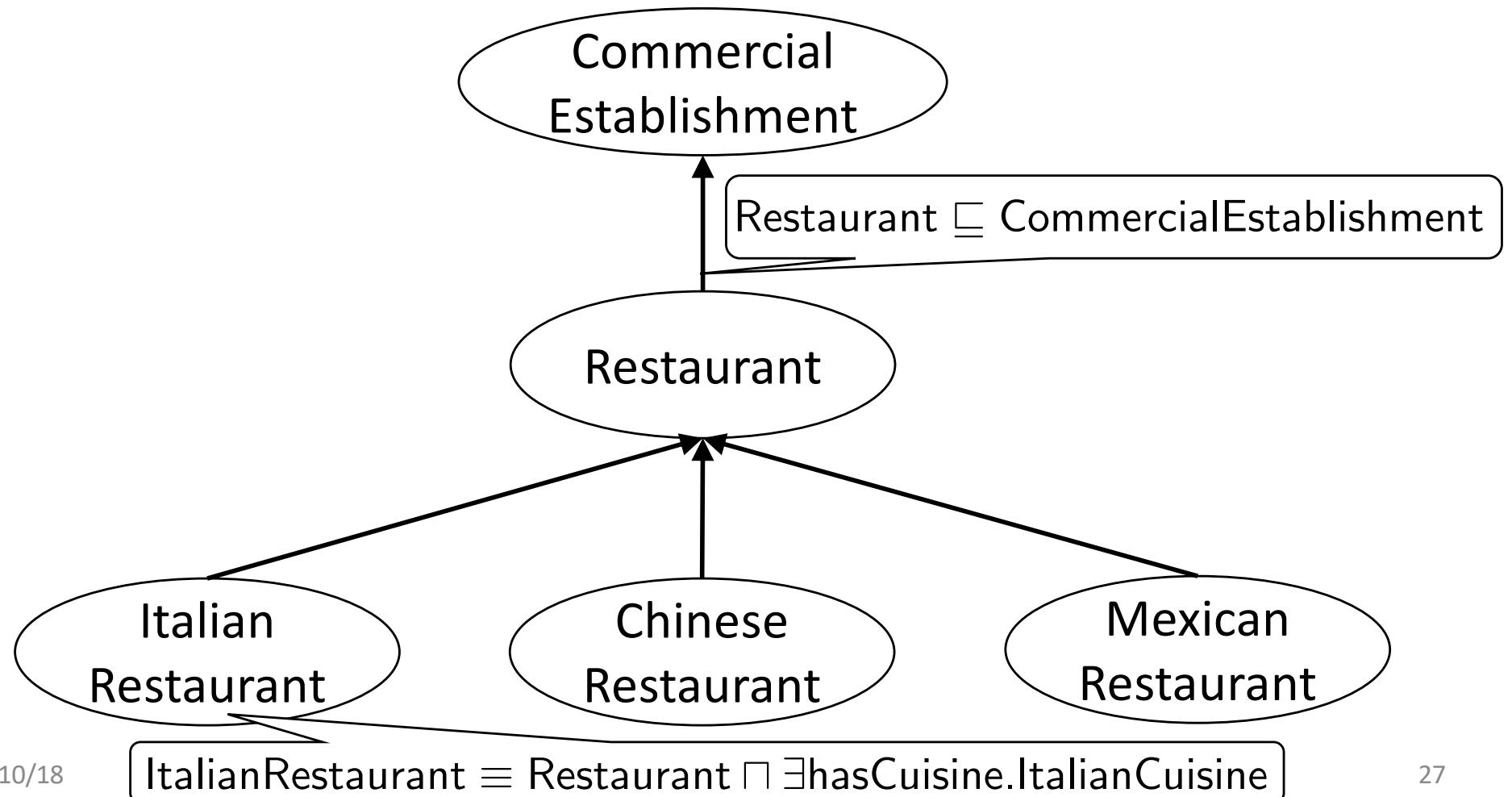
- 述語が表現するイベントの時間的性質は以下の4クラスに細文類される [Vendler 1967]
 - Stative**: 所与の時間における属性や状態
I know my departure gate.
 - Activity**: 動作の完了を含まないイベント
John is flying.
 - Accomplishment**: 動作の完了とその結果を含むイベント
Sally booked her flights.
 - Achievement**: Accomplishment で動作を伴わなもの
(Accomp. とまとめて telic eventualities)
She found her gates.

記述論理 (Description Logics; DL)

- 実世界の形式化であるモデルのドメインの記述に特化した一階述語論理のサブセット
オブジェクト間の概念ネットワークを表現
 - **terminologies**: オブジェクトを汎化したカテゴリ(概念)
 - 等価な一階述語論理に変換可能
- 記述論理により記述された実世界(知識ベース)
 - TBox: terminology の集合 (概念間階層構造: **ontology**)
 - Abox: 具体物に関する事実の集合
- 利用例: Semantic Web [Fensel+ 2003]
記述論理に基づく **Web Ontology Language (OWL)** [McGuiness+ 2004] を用いたWebコンテンツ記述のための意味論

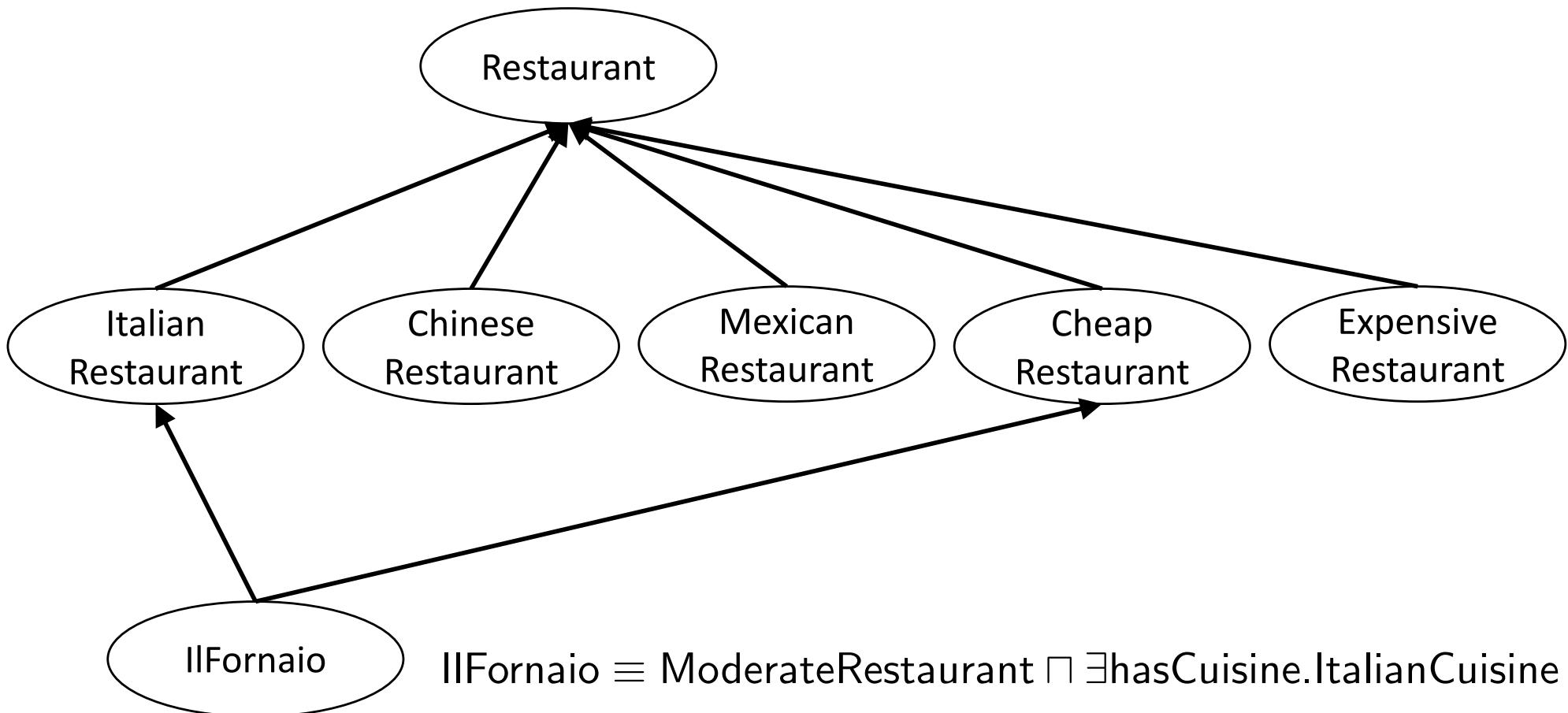
記述論理: 例

- レストランドメインの Tbox に関する ontology
 - 概念間の包含関係をグラフ表現



記述論理: 推論

- Instance checking:
概念ネットワークにおける包含関係を辿って推論

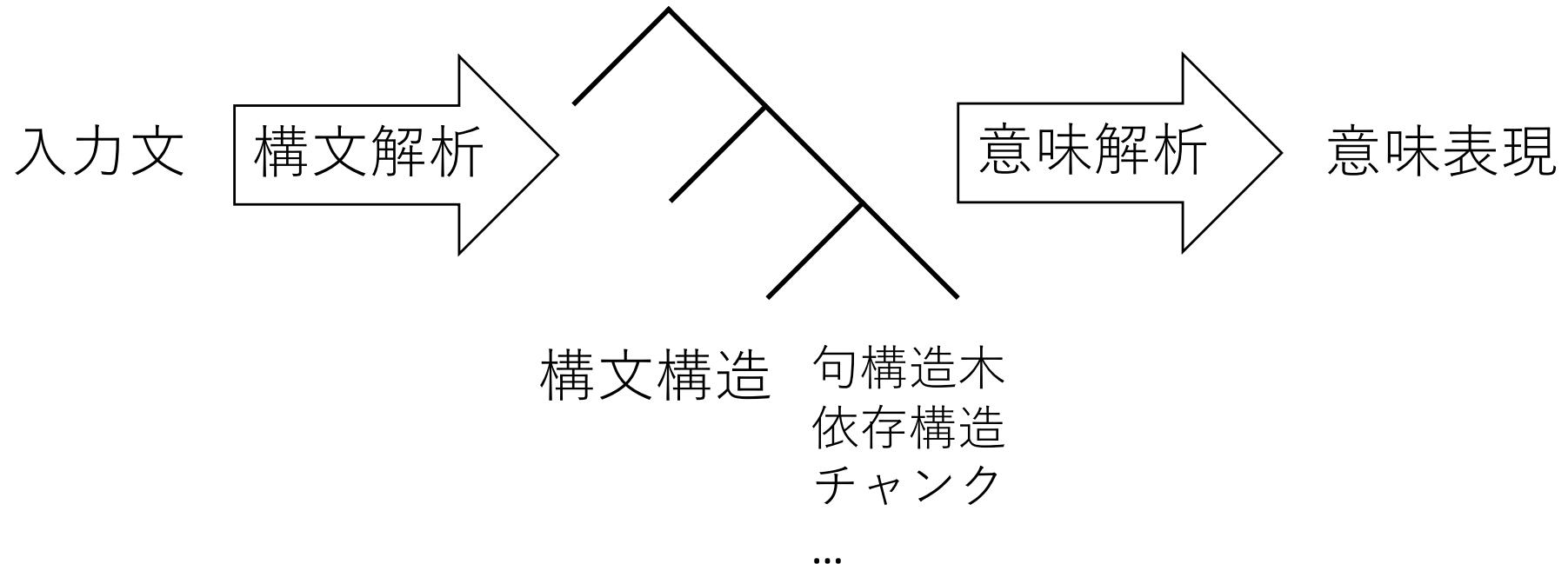


意味の計算

文の意味表現をどう計算するか？

構成的意味論

- 構成的意味論: 文の意味をその要素(単語)から計算
 - 形式文法の文法や語彙に意味の計算方法を付与し計算

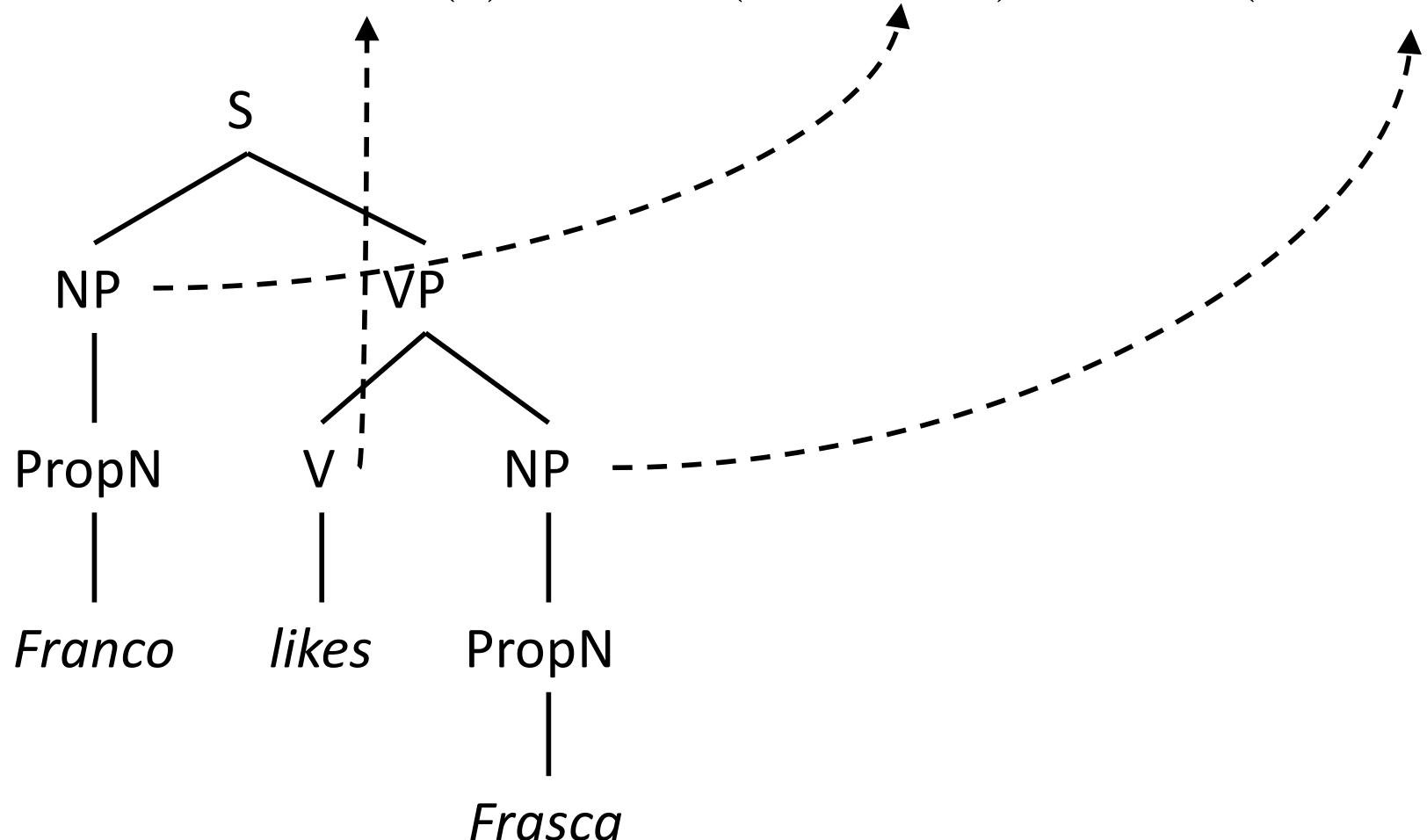


- 知識ベースとの照合に基づく質問応答システムに必要

意味解析: 例

- *Franco likes Frasca*

$$\exists e \text{ Liking}(e) \wedge \text{Liker}(e, \text{Franco}) \wedge \text{Liked}(e, \text{Frasca})$$



構成的意味論に基づく意味解析: 文法規則に対する意味計算の付与

- CFG の生成規則に意味の計算を割り当てる

$$A \rightarrow \underline{\alpha_1 \dots \alpha_n} \qquad \qquad \underline{\{f(\alpha_j.sem, \dots, \alpha_k.sem)\}}$$

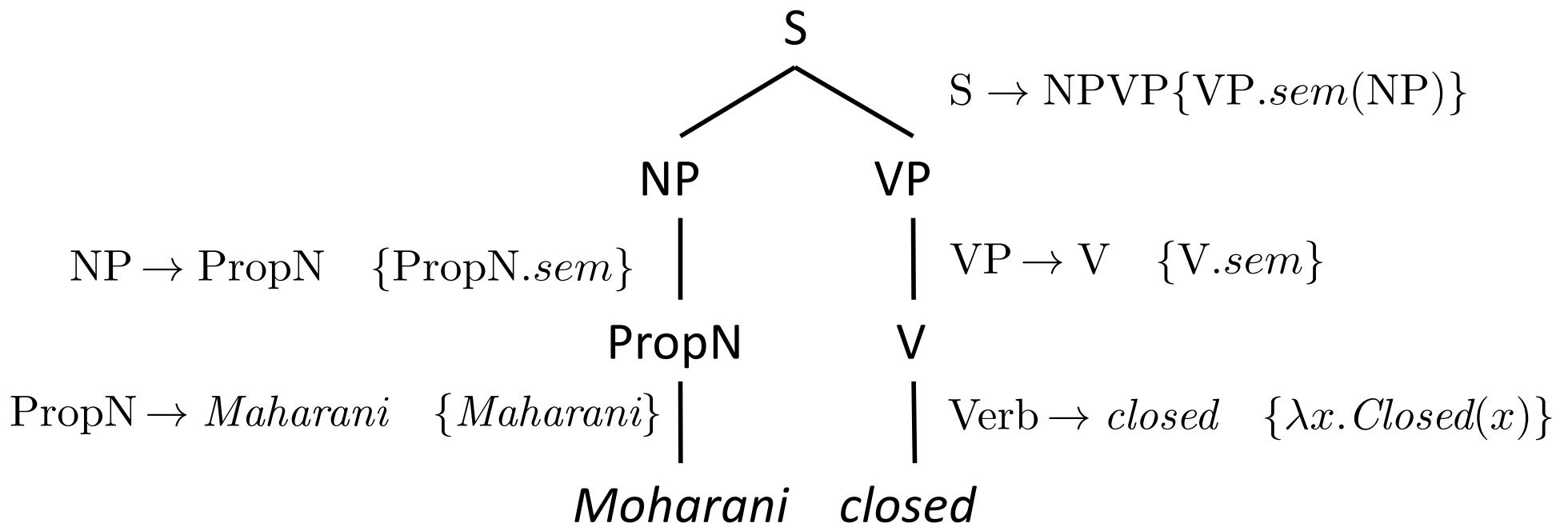
Aの構成素 A.sem

- A.sem を A の構成素 α_i の意味表現 $\alpha_i.sem$ に関数 f を適用することで得る
 - 関数 f は一階述語論理とラムダ計算により定義

構成的意味論に基づく意味解析: 例

Moharani closed. (イベント変数なし)

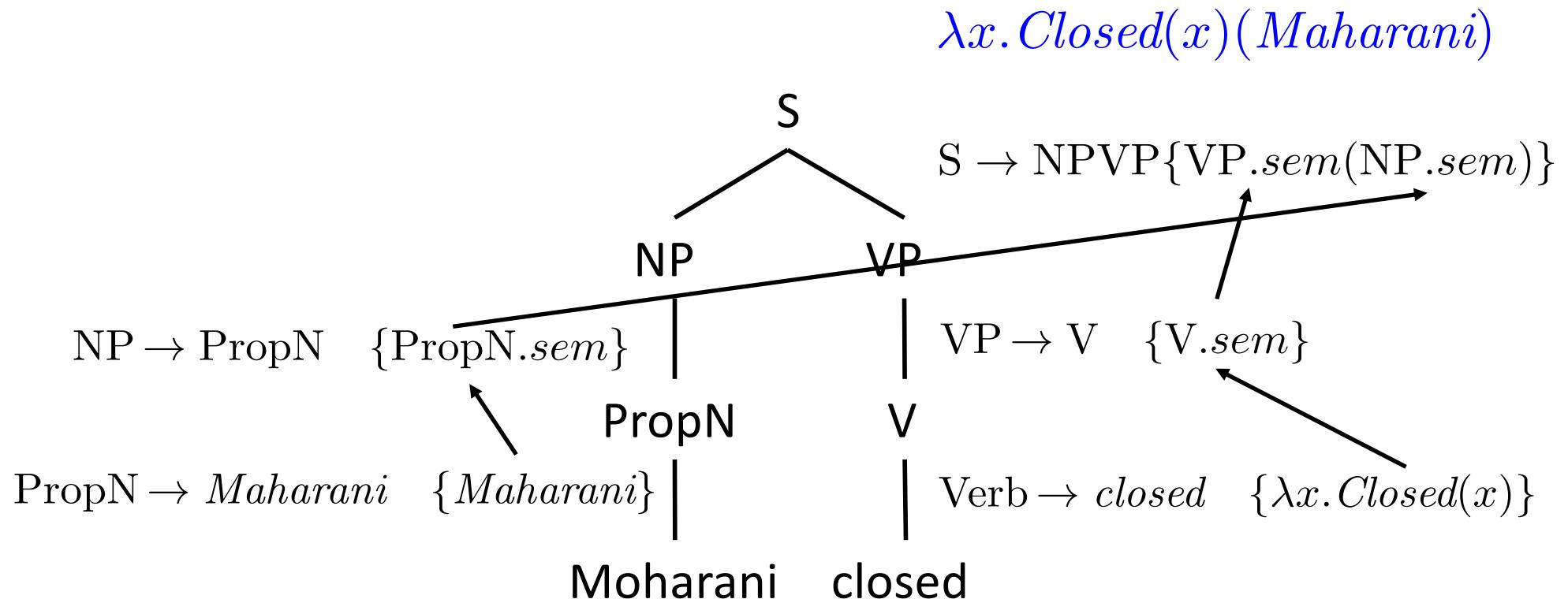
- 意味表現: $Closed(Maharani)$



構成的意味論に基づく意味解析: 例

Moharani closed. (イベント変数なし)

- 意味表現: $\text{Closed}(\text{Maharani})$



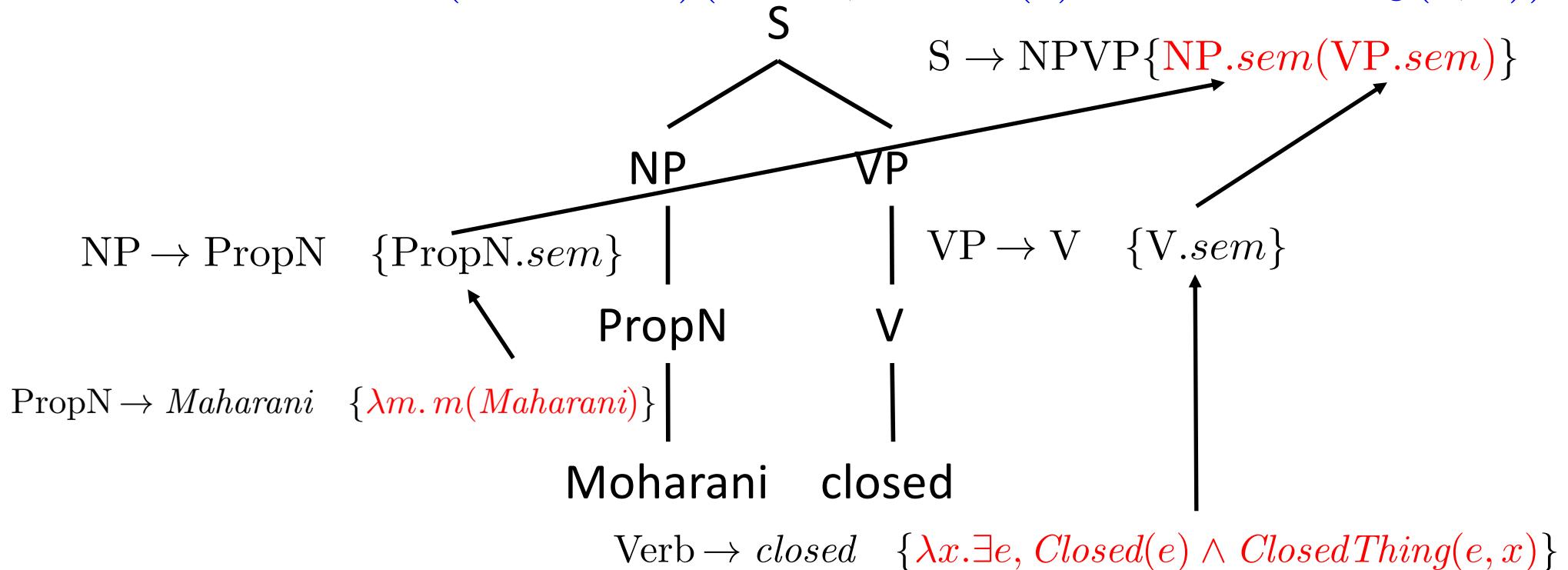
構成的意味論に基づく意味解析: 例

Moharani closed. (イベント変数あり)

- 意味表現: $\exists e \ Closed(e) \wedge ClosedThing(e, Maharani)$

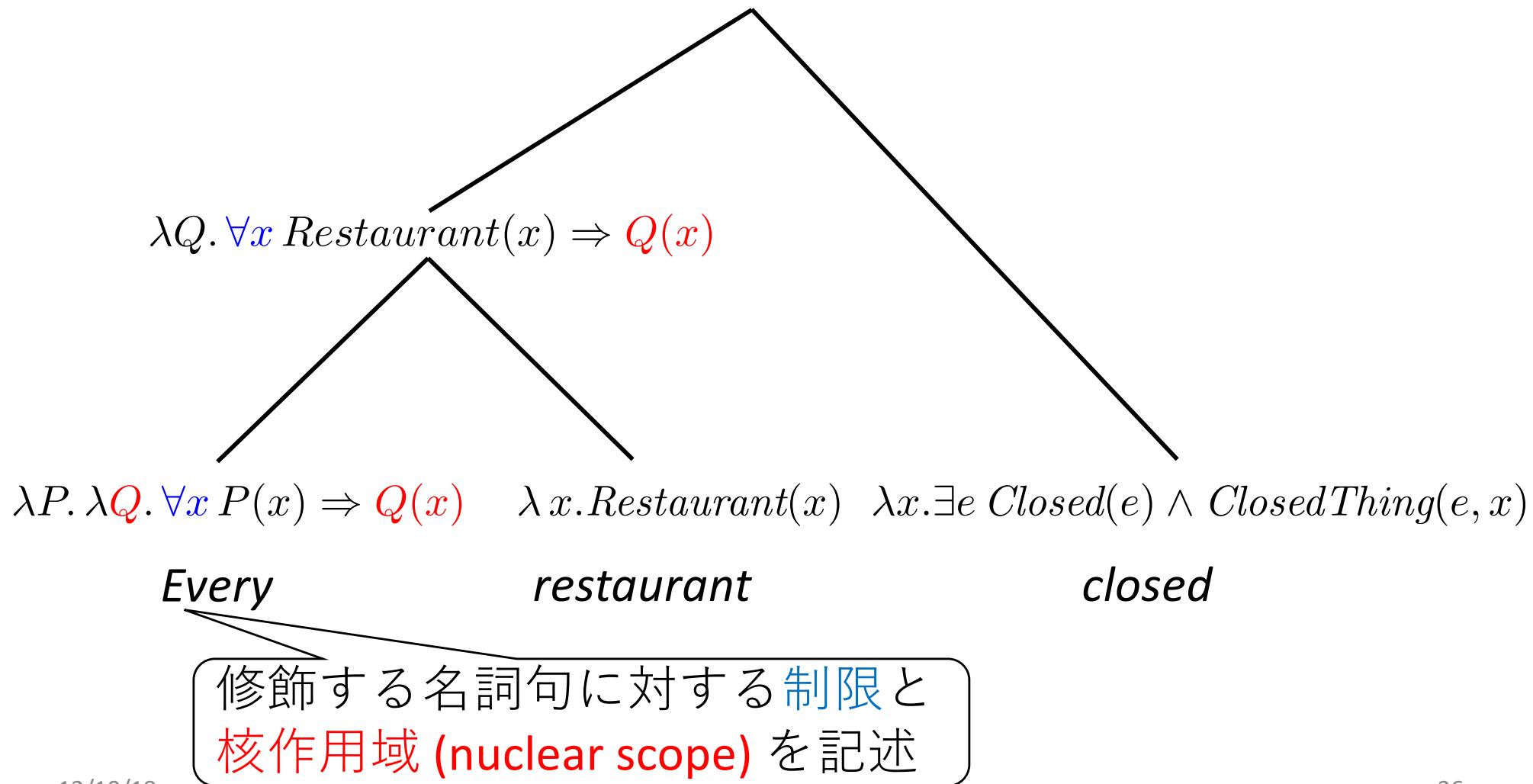
$\lambda x. \exists e, Closed(e) \wedge ClosedThing(e, x)(Maharani)$

$\lambda m. m(Maharani)(\lambda x. \exists e, Closed(e) \wedge ClosedThing(e, x))$



構成的意味論に基づく意味解析: 例
Every restaurant closed.

- 意味表現: $\forall x \text{ Restaurant}(x) \Rightarrow \exists e \text{ Closed}(e) \wedge \text{ClosedThing}(e, x)$

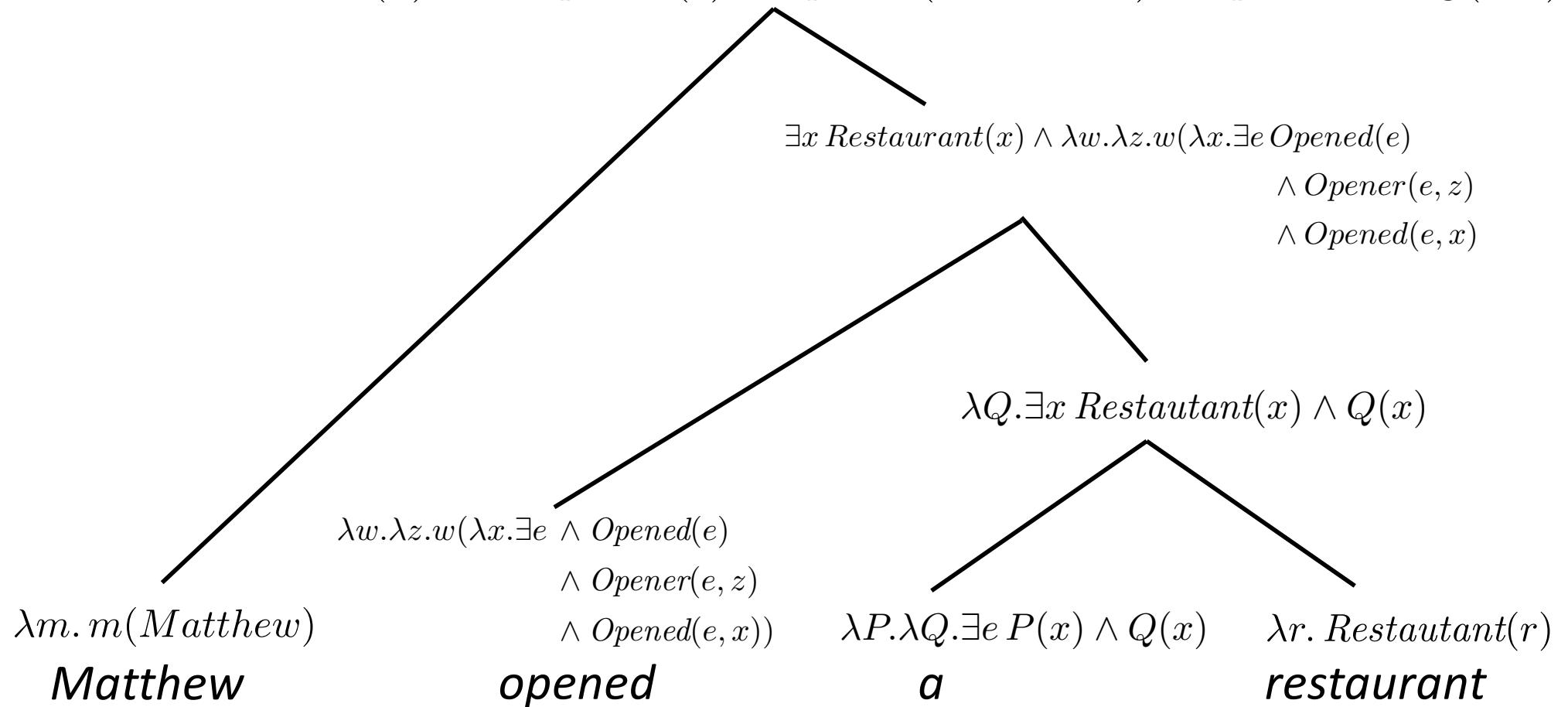


構成的意味論に基づく意味解析: 例

Matthew opened a restaurant

- 意味表現:

$$\exists x \text{ Restaurant}(x) \wedge \exists e \text{ Opened}(e) \wedge \text{Opener}(e, \text{Matthew}) \wedge \text{OpenedThing}(e, x)$$



構成的意味論に基づく意味解析: まとめ

- 語彙項目にラムダ表現(関数)を割り当てる
- 分岐がなければ子ノードの意味表現を親にコピー
- 分岐があれば λ -reduction によりある子の意味表現(関数)をそれ以外の子の意味表現に適用する

構文解析時に適用する各文法規則に対応した
意味の計算を行うことで文の意味を計算可能

量化子のスコープの曖昧性

- Every boy loves a girl の意味表現は二通り

- 女の子が一人

$$\forall x Boy(x) \Rightarrow \exists y Girl(y) \wedge \exists e Loving(e) \wedge Lover(e, x) \wedge Loved(e, y)$$

- 女の子が男の子ごとに一人

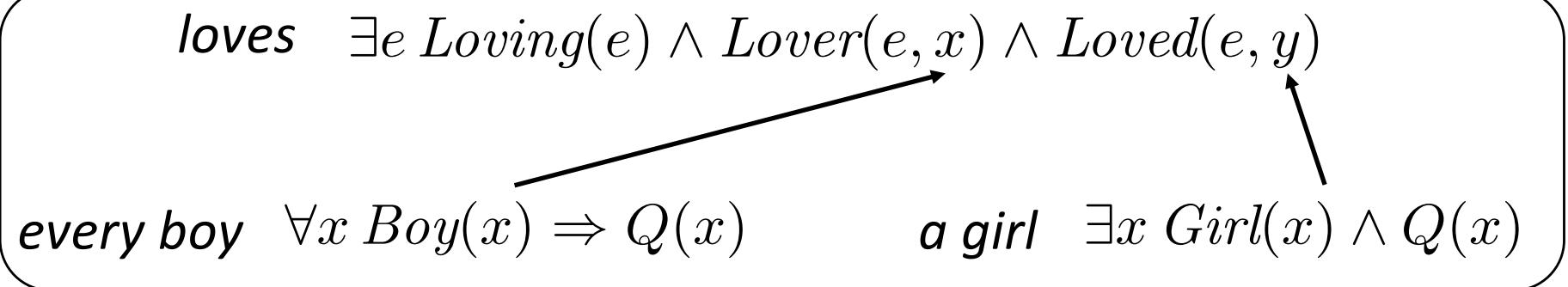
$$\forall y Girl(y) \wedge \exists x Boy(x) \Rightarrow \exists y \exists e Loving(e) \wedge Lover(e, x) \wedge Loved(e, y)$$

- 量化子のスコープの曖昧性にどう対処するか？

- 量化子のスコープの曖昧性を保持した意味表現を設計
 - 上記の意味表現から可能な解釈を生成する
 - 可能な解釈の中から妥当な解釈を得る

量化子のスコープの曖昧性: 対策 Cooper storage [Cooper 1983] (1/2)

- 構文木のノードに (評価を遅延した) 意味表現を割り当てる(store)

$$\exists e \text{ Loving}(e) \wedge \text{Lover}(e, s_1) \wedge \text{Loved}(e, s_2)$$
$$(\lambda Q. \forall x \text{ Boy}(x) \Rightarrow Q(x), 1)$$
$$(\lambda Q. \exists x \text{ Girl}(x) \wedge Q(x), 2)$$


- 最終的な意味表現で、あり得る適用順序を全適用 (retrieve)

量化子のスコープの曖昧性: 対策 Cooper storage [Cooper 1983] (2/2)

- Cooper storage の限界

Every student did not graduate.

$$\neg(\forall x \text{ Student}(x) \Rightarrow \exists e \text{ Graduating}(e) \wedge \text{Graduated}(e, x))$$

$$\forall x \text{ Student}(x) \Rightarrow \neg(\exists e \text{ Graduating}(e) \wedge \text{Graduated}(e, x))$$

- Cooper storage は否定を扱えない
- 評価を遅延した意味表現に関する可能な意味計算の順序について、制約をモデル化することができない

量化子のスコープの曖昧性: 対策 hole 意味論 [Bos 1996]

- 評価を遅延した意味表現について,
可能な計算の順序に関する制約をモデル化

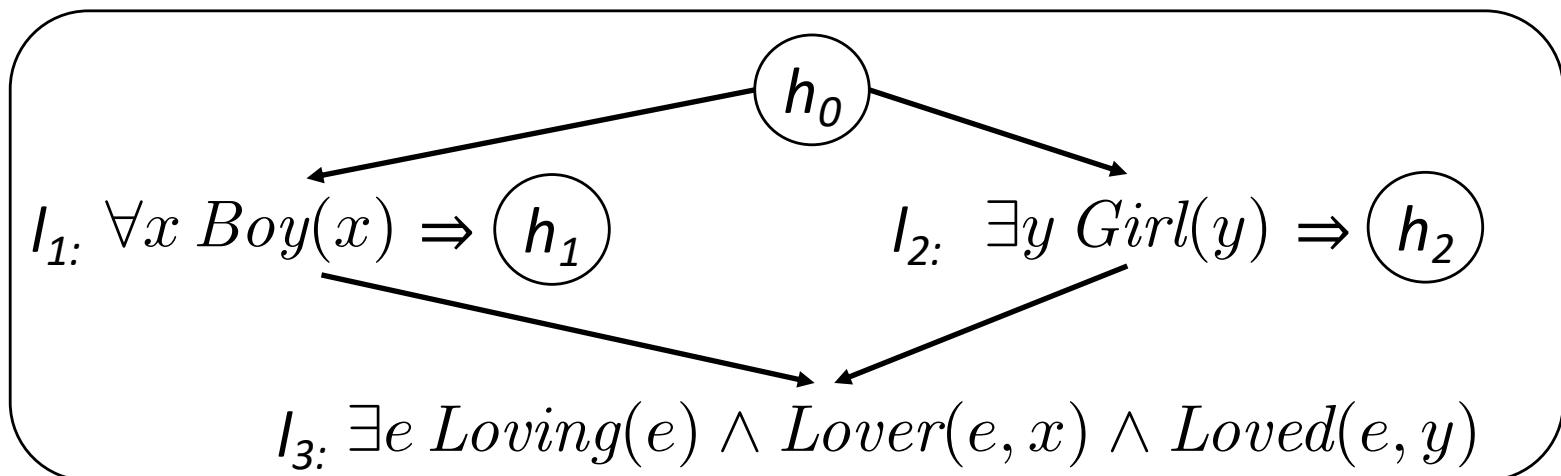
Every boy loves a girl.

$$l_1 : \forall x Boy(x) \Rightarrow h_1$$

$$l_2 : \forall y Girl(y) \wedge h_2$$

$$l_3 : \exists e Loving(e) \wedge Lover(e, x) \wedge Loved(e, y)$$

$$l_1 \leq h_0, l_2 \leq h_0, l_3 \leq h_1, l_3 \leq h_2$$



イディオムと構成性 (1/2)

- 意味が構成的でない イディオム の意味表現をどのように扱うか?
例)

kick the bucket = *die*

the tip of the iceberg = *the beginning*

- ナイーブな解決策: 単語列に意味表現を与える

VP → *kick the bucket* {*Die*}

NP → *the tip of the iceberg* {*Beginning*}

- 注) これだけでは不十分

The 10 employees represent the merest tip of the iceberg.

イディオムと構成性 (2/2)

- 意味が構成的でない イディオム の意味表現をどのように扱うか？
例)

kick the bucket = die

the tip of the iceberg = the beginning

- イディオムのための汎化規則:
 $NP \rightarrow tipNP\ of\ the\ icebergNP \quad \{Beginning\}$
 - 文法規則に語彙項目を混ぜる
 - イディオムに特化した構造を認める
 - どの構成素にも関係のない述語の導入を認める

文の数理的意味表現

ベクトル意味論に基づく文の意味表現

- 文の意味 = 実数値ベクトル (埋め込み)
 - 単純なベクトル演算に基づく文の意味表現 [Michell+ 2008]
 - 学習したベクトル演算に基づく文の意味表現 [Le+ 2014, Logeswaran+ 2018]
 - 文・段落ベクトルの直接学習 [Le+ 2014]
- 一階述語論理(述語 = 高階関数)を意識した意味計算
 - 形容詞の意味表現 = 行列 [Baroni+ 2010]
 - 動詞の意味表現 = テンソル [Grefenstette+ 2011]

正解の文ベクトルは与えられないので
文の類似性判定や応用タスクで評価

学習したベクトル演算に基づく文の意味表現

[Michell+ 2008]

- 単語の意味表現(ベクトル)を単純なベクトル演算で合成し、句の意味表現(ベクトル)を計算
 - 加算: $p_i = u_i + v_i$
 - 乗算: $p_i = u_i \cdot v_i$
- 実験: 句 (主語(名詞)-述語(自動詞)) の類似性判定
 - BNC コーパスから抽出した句に対し動詞を類義語で置換
 - 句の意味が近い類義語 (high) と遠い類義語 (low) を選ぶ
例) The child *strayed* (high: *roamed*; low: *digressed*)

Model	high	low	ρ
動詞のみ	0.27	0.26	0.08
加算	0.59	0.59	0.04
乗算	0.42	0.28	0.17
human	4.94	3.25	0.40

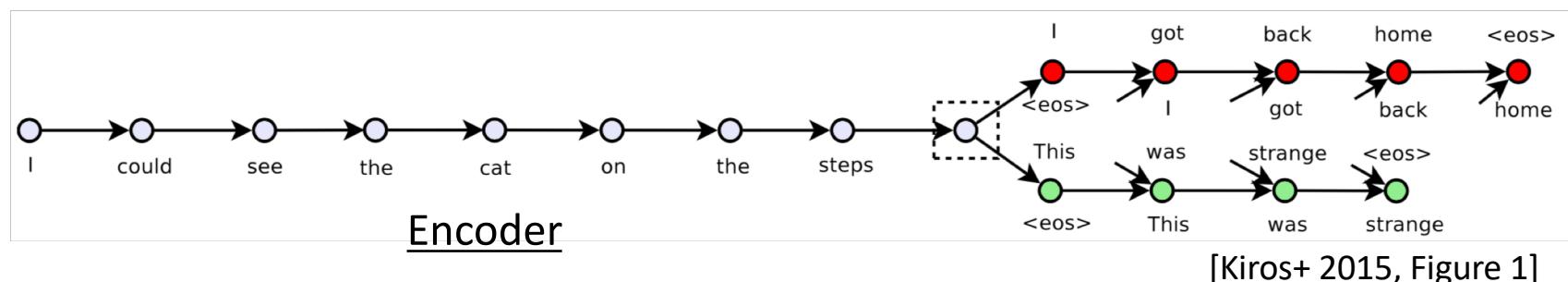
コサイン類似度

置換した句の類似度
を1-7で付与

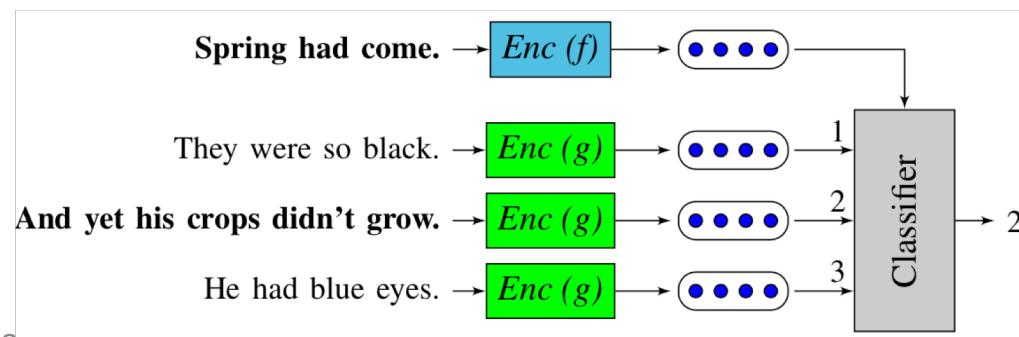
単語の意味表現に対するベクトル演算の学習

[Socher+ 2011, Kiros+ 2015, Logeswaran+ 2018]

- 教師なしタスクを通じて単語ベクトルの合成関数をニューラルネットワーク (RNN) 等で学習
 - Autoencoder [Socher+ 2011]: 入力文を復元
 - Skip-thoughts [Kiros+ 2015]: 前後の文を生成



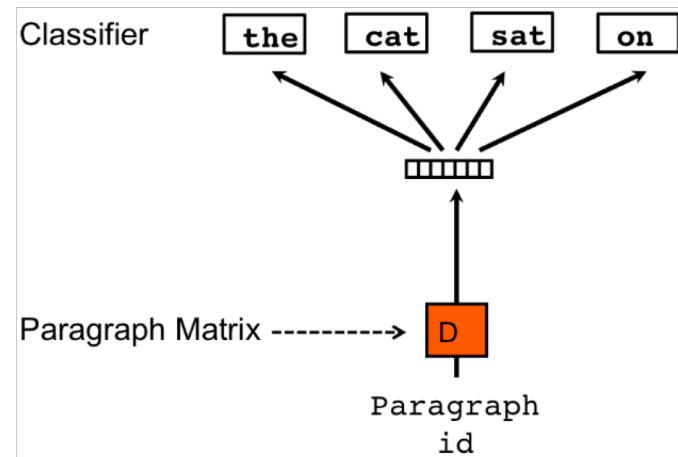
- Quick thoughts [Logeswaran+ 2018]: 前後の文を文集合から選択



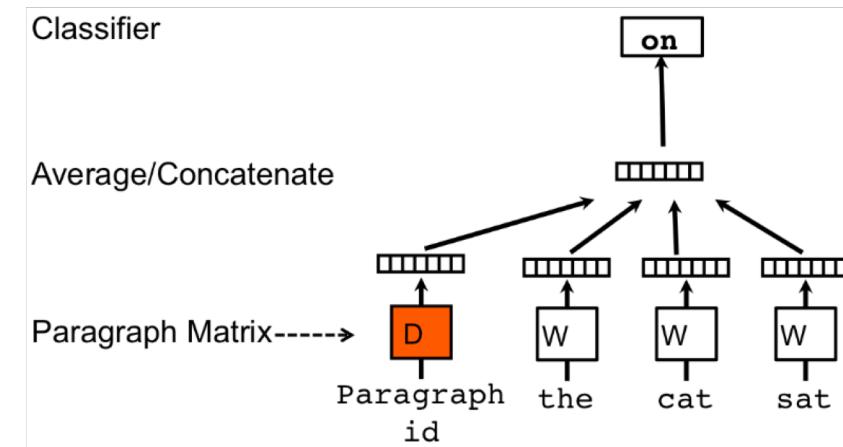
重い softmax 関数を計算する
decoder がないため高速

文・段落ベクトルの直接学習 [Le+ 2014]

- 文のID(または文と周辺単語のID)からその文に含まれる単語を予測する問題を解く



[Le+ 2014, Figure 3]



[Le+ 2014, Figure 2]

- 未知の文を扱うには再学習が必要
 - 単語の埋め込みは固定して勾配計算

文のベクトル表現の評価実験 [Logeswaran+ 2018]

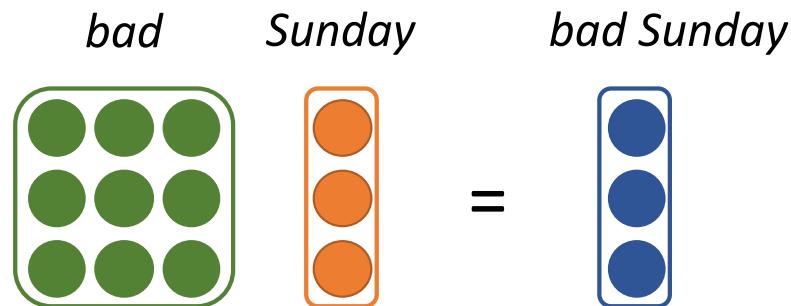
- GloVe BoW: 単語ベクトルの総和
- SDAE (Sentence De-noising AutoEncoder)
- Combine-skip (skip-thoughts): uni-skip ; bi-skip
- Combine-QT (quick thoughts): uni-QT ; bi-QT

モデル	次元数	学習	評判分析				質問 分類	換言同定		意味的関連度		
			MR	CR	SUBJ	MPQA		TREC	MSRP (Acc)	MSRP (F1)	SICK r	SICK ρ
GloVe BoW	300	-	78.1	80.4	91.9	87.8	85.2	72.5	81.1	0.764	0.687	0.425
SDAE	2400	192h	67.6	74.0	89.3	81.3	77.6	76.4	83.4	n/a	n/a	n/a
ParagraphVec	<500	4h	61.5	68.6	76.4	78.1	55.8	73.6	81.9	n/a	n/a	n/a
Combine-skip	4800	336h	76.5	80.1	93.6	87.1	92.2	73.0	82.0	0.858	0.792	0.269
Combine-QT	4800	11h	78.2	84.4	93.3	88.0	90.8	76.2	83.5	0.860	0.796	0.267

一階述語論理を意識した意味計算

[Baroni+ 2010, Grefenstette+ 2011]

- 形容詞 = 名詞ベクトルへの線形変換(行列) [Baroni+ 2010]
 - コーパスから計算した形容詞句のベクトルを正解とし、部分的最小二乗回帰タスクを解く



- 動詞 = 高階テンソル [Coecke+ 2010]
 - 主語・目的語ベクトルのクロネッカー積の総和により、動詞の意味をテンソルとして計算
 - 他動詞の句のベクトル表現で乗算 [Mitchell+ 2008] を上回る

まとめ

- 文の記号的意味表現: 文の意味 = 記号の組み合わせ
 - 意味表現に求められる要件
 - 記号的意味表現: 一階述語論理, 記述論理
 - 構成的意味論: 構文構造に基づく意味の計算
 - 課題: 量化子のスコープの曖昧性, イディオム(非構成性)
- 文の数理的意味表現: 文の意味 = ベクトル
 - ベクトル意味論に基づく文の意味表現 = ベクトル
 - ベクトル演算に基づく文の意味表現の合成
 - 文・段落ベクトルの直接学習
 - 課題: 複雑な意味的関係の計算, 文の意味の情報量の差