Lecture 4: Truth-Conditional Semantics

Weiwei Sun

Department of Computer Science and Technology
University of Cambridge

Michaelmas 2024/25

Lecture 4: Truth-Conditional Semantics

1. Functions and λs
2. Truth conditions
3. First-Order Predicate Logic
4. Davidsonian semantics

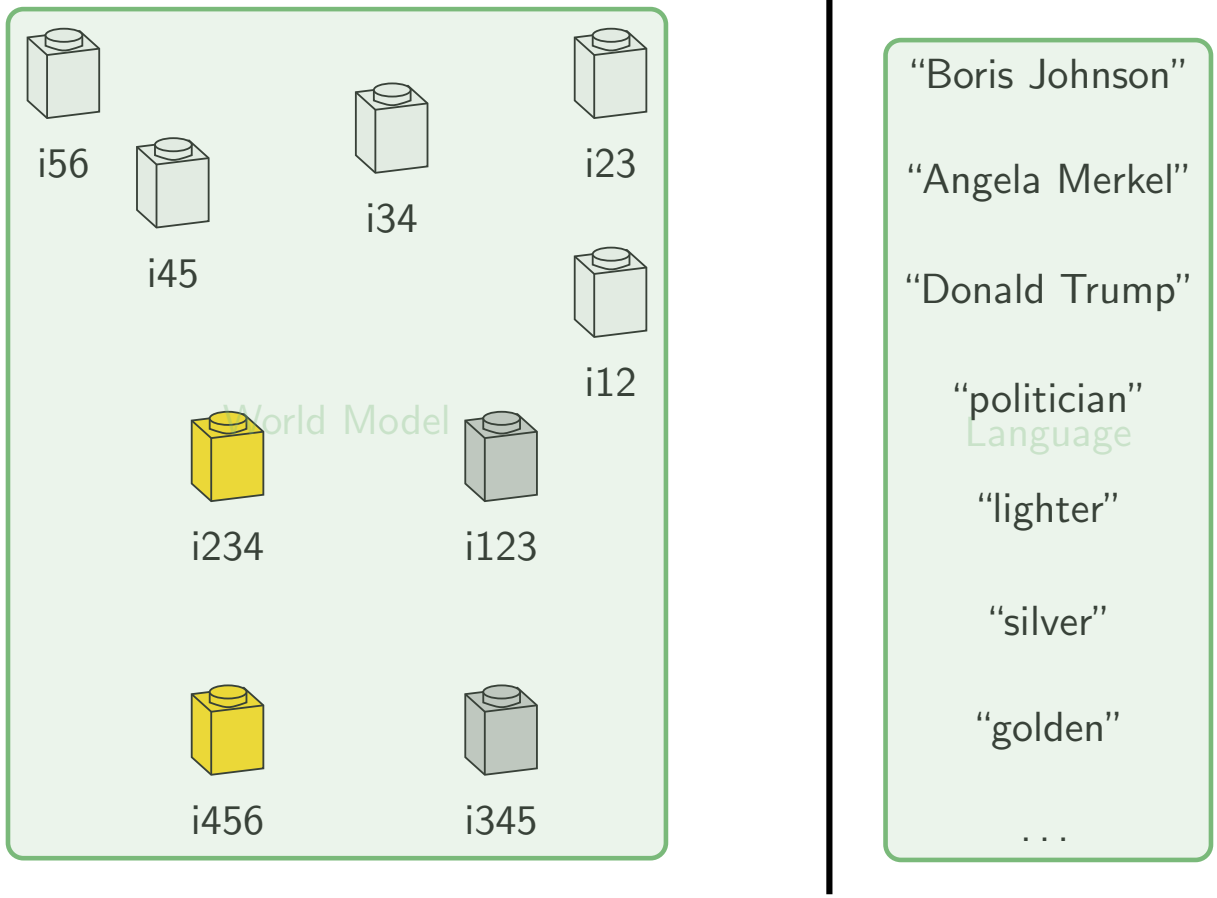
Assignment I

Using predicate logic, write the truth-conditional semantic representations of the following sentences.

- London Tube drivers to strike over pay
- We don't want to go on strike
- Aslef is seeking a pay agreement with London Underground.
- No trade union can accept any pay proposal where management.
- about 10,000 of its members were involved in the dispute.
- The cost of living is always important to our finances.
- the rise is likely to be £10.50 to £628 a month
- Someone receiving attendance allowance will see an increase of about £1.85 a week in April.
- Why the benefit rise could have been higher
- Many people face higher monthly repayments

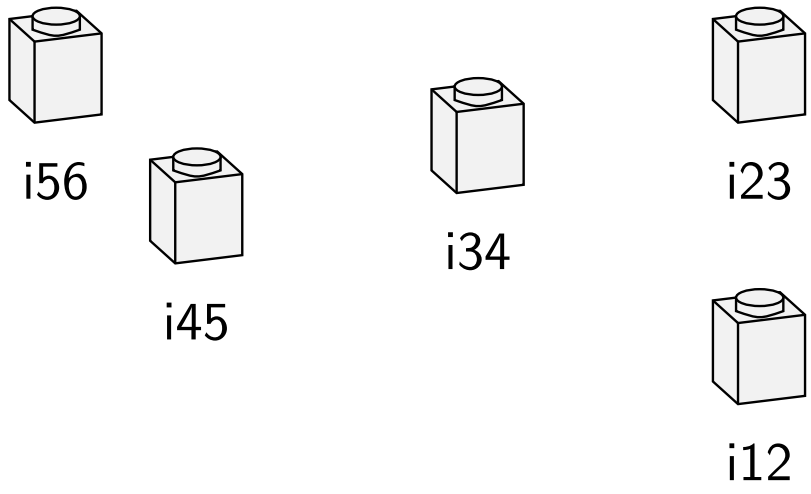
Functions and λs

Set-theoretic semantics



- The semantics is defined in terms of Set Theory.
- $[\![\mathit{Trump}]\!] = i12$
- How can we easily and precisely describe sets as well as operations over sets?

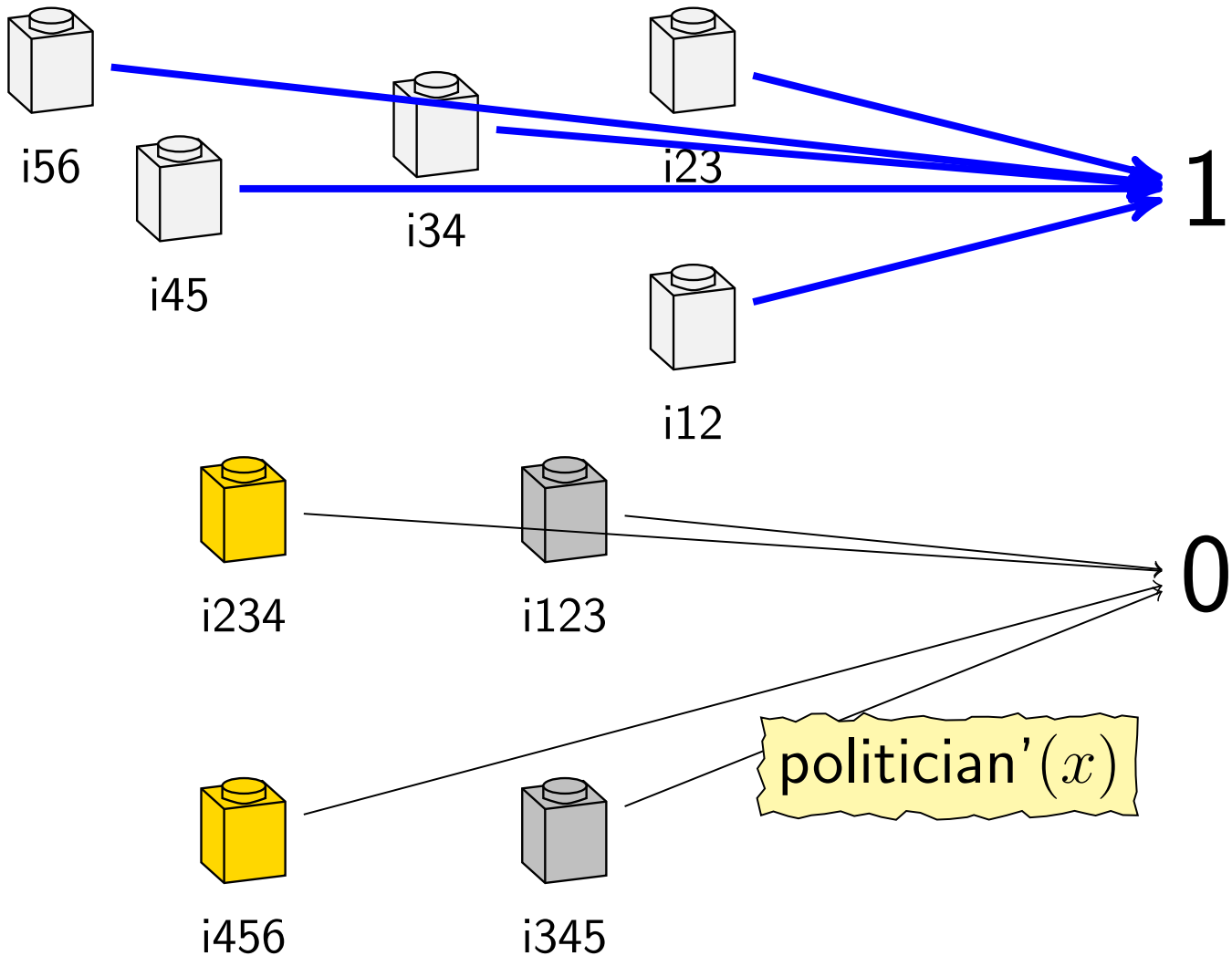
Buckets/sets $\rightarrow$ functions



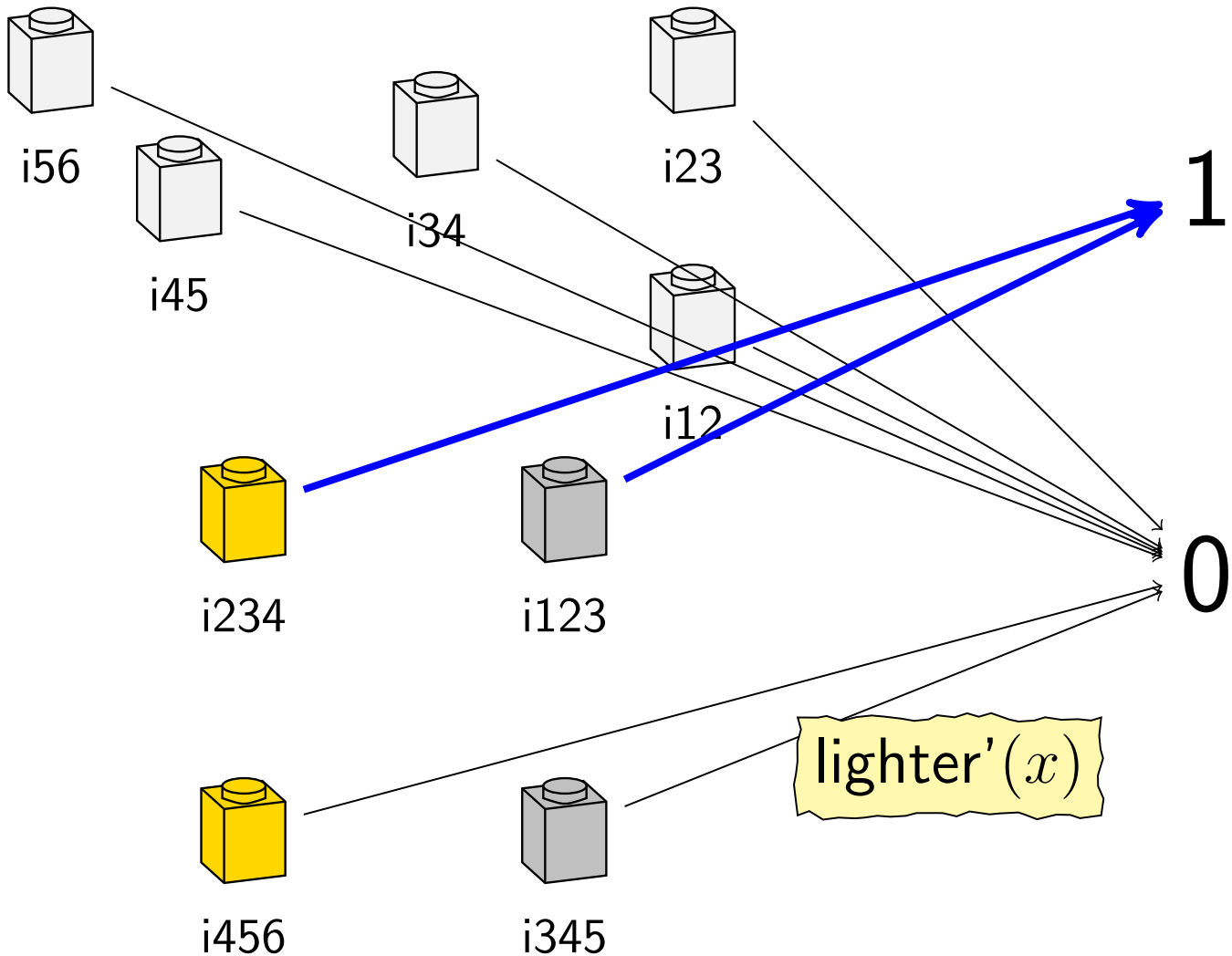
1

0

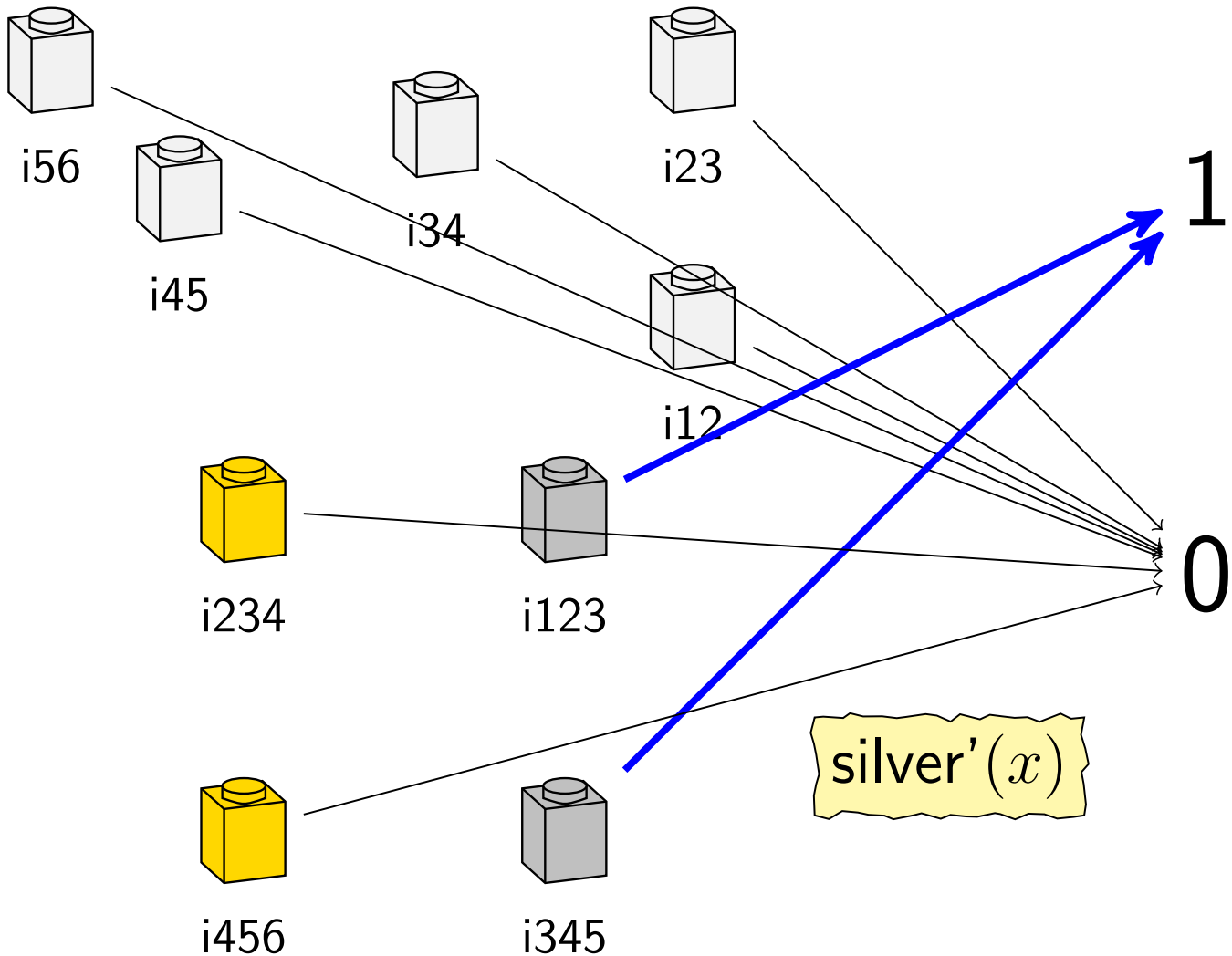
Buckets/sets $\rightarrow$ functions



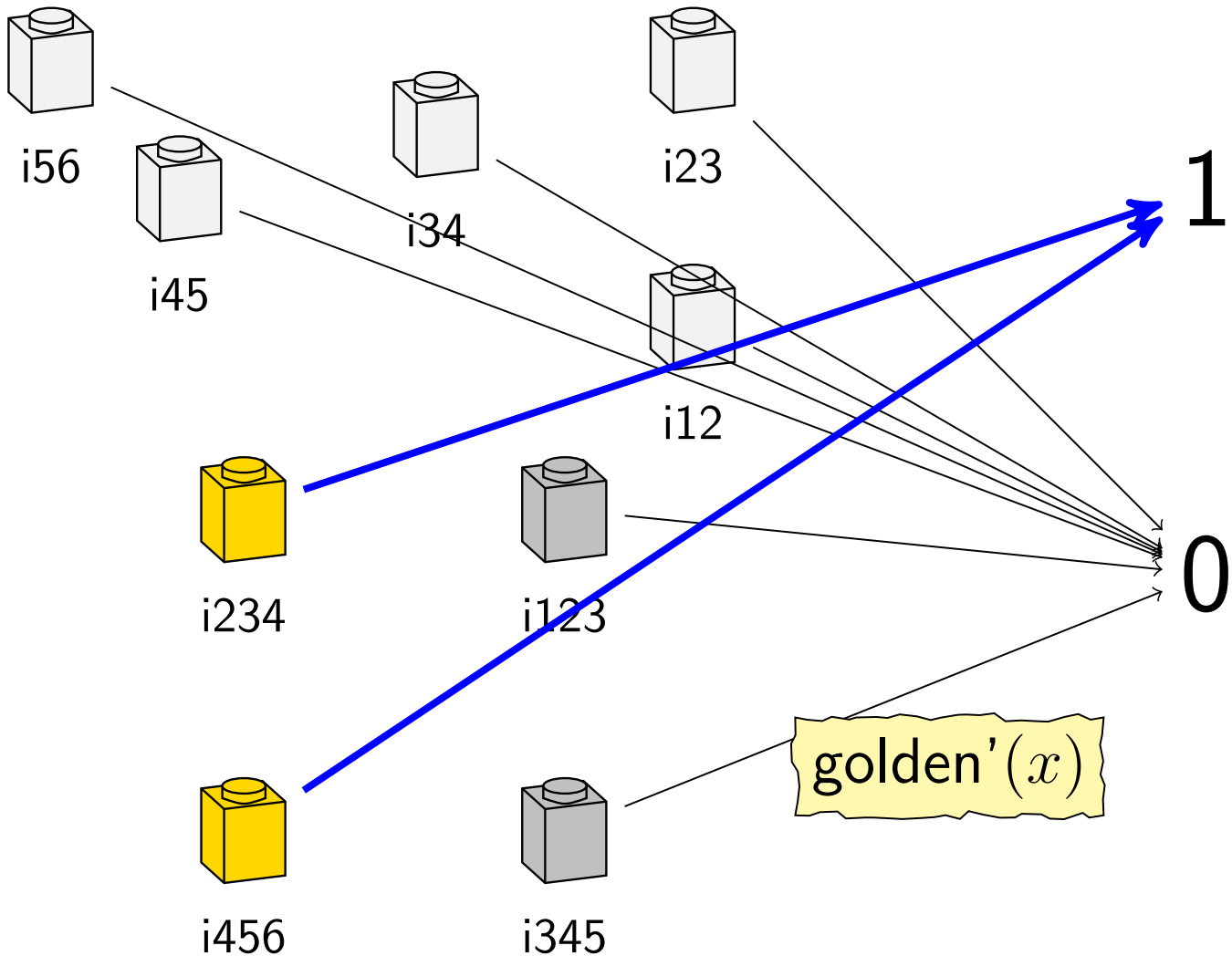
Buckets/sets $\rightarrow$ functions



Buckets/sets $\rightarrow$ functions



Buckets/sets $\rightarrow$ functions



Predicates are functions; predicates are sets.

Q What is the meaning of *politician*?

A politician'

- politician' is a semantic predicate which is a set and also a function.
- Discourse referents are mapped to either 0 or 1 through politician'. The referents mapped to 1 indicate politicians.
- How can we easily and precisely describe sets as well as operations over sets?

  It is a great idea to define functions with a minimal programming language — λ-calculus.

Building functions

λ-calculus — a simple notation for functions and application

- β-reduction/function application:

$$[\lambda x.M](N) \longrightarrow M[x := N]$$

- Apply a λ-term to an argument, and get a value.

More online: `https://plato.stanford.edu/entries/lambda-calculus/`

Example

λ-calculus allows us to build functions in a very convenient way.

- $f(x) = x^2 \longleftrightarrow [\lambda x.[x^2]]$
- $f(5) = 25 \longleftrightarrow [\lambda x.[x^2]](5) = 25$
- $g(x, y) = x^2 + y^2 \longleftrightarrow [\lambda x.[\lambda y.[x^2 + y^2]]]$
- $g(2, 1) = 5 \longleftrightarrow [\lambda x.[\lambda y.[x^2 + y^2]]](2)(1) = 5$

Challenge: Forms such as $f(f(f(f(\ldots))))$ are annoying.

Simple types

From a nonempty set $\mathbf{BasTyp}$ of *basic types*, the set $\mathbf{Typ}$ is the smallest set such that

- $\mathbf{BasTyp} \subseteq \mathbf{Typ}$,
- $\langle \sigma, \tau \rangle \in \mathbf{Typ}$ if $\sigma, \tau \in \mathbf{Typ}$.

A type of the form $\langle \sigma, \tau \rangle$ is said to be a *functional type*.

Example

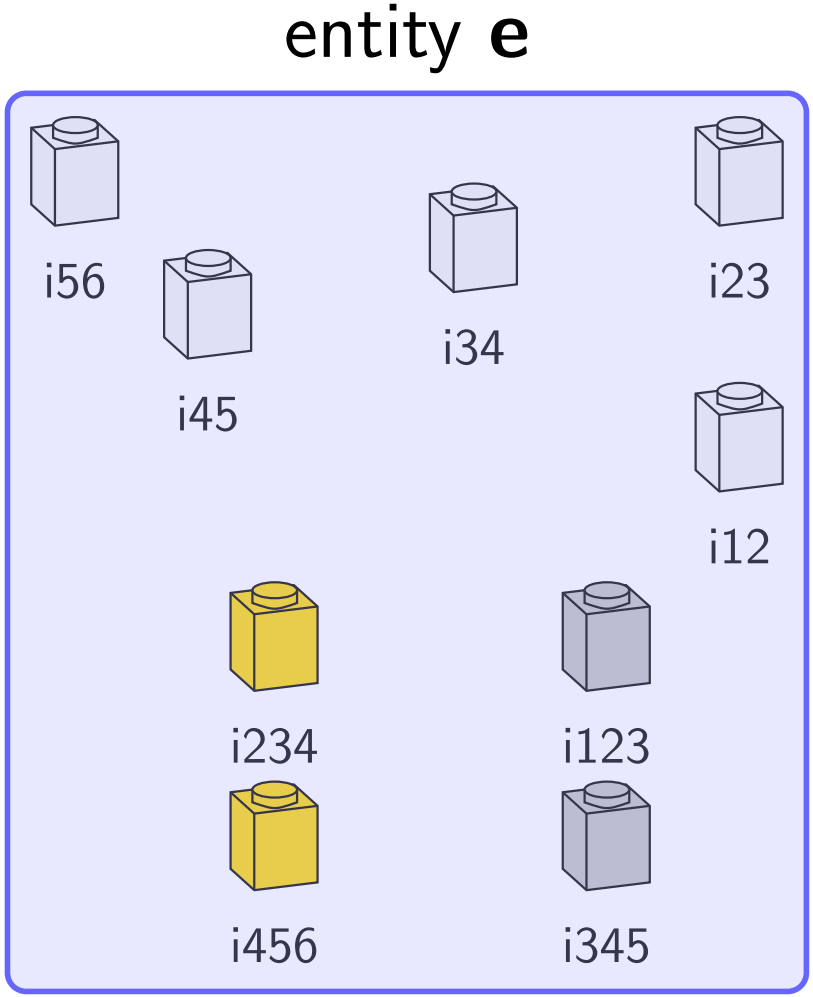
- Assume $\mathbf{e}$ for individuals and $\mathbf{t}$ for `true`/`false`,
- then $\langle \mathbf{e}, \mathbf{t} \rangle$ is the type for unary relations,
- and $\langle \langle \mathbf{e}, \mathbf{t} \rangle, \langle \mathbf{e}, \mathbf{t} \rangle \rangle$ is for the type of a function mapping unary relations into unary relations.

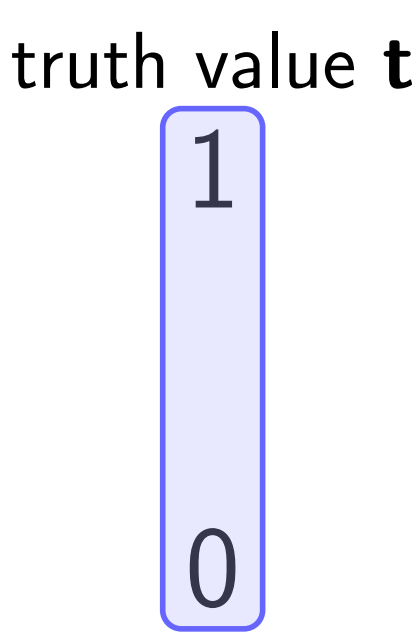
`C/C++/Java/Typescript` vs `Python/Javascript`

e, **t** and **e** to **t**

Gottlob Frege

There are only two atomic things, truth values and individuals. All other things are created by function application.



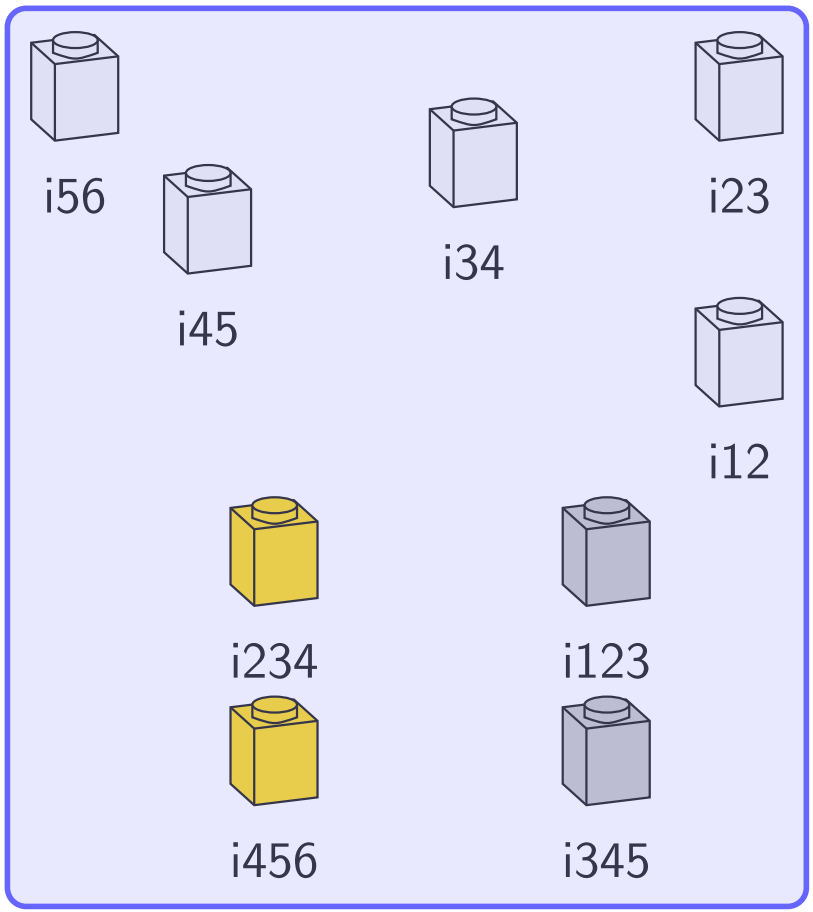


e, **t** and **e** to **t**

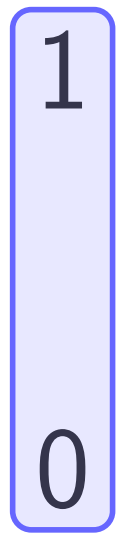
Gottlob Frege

There are only two atomic things, truth values and individuals. All other things are created by function application.

entity **e**

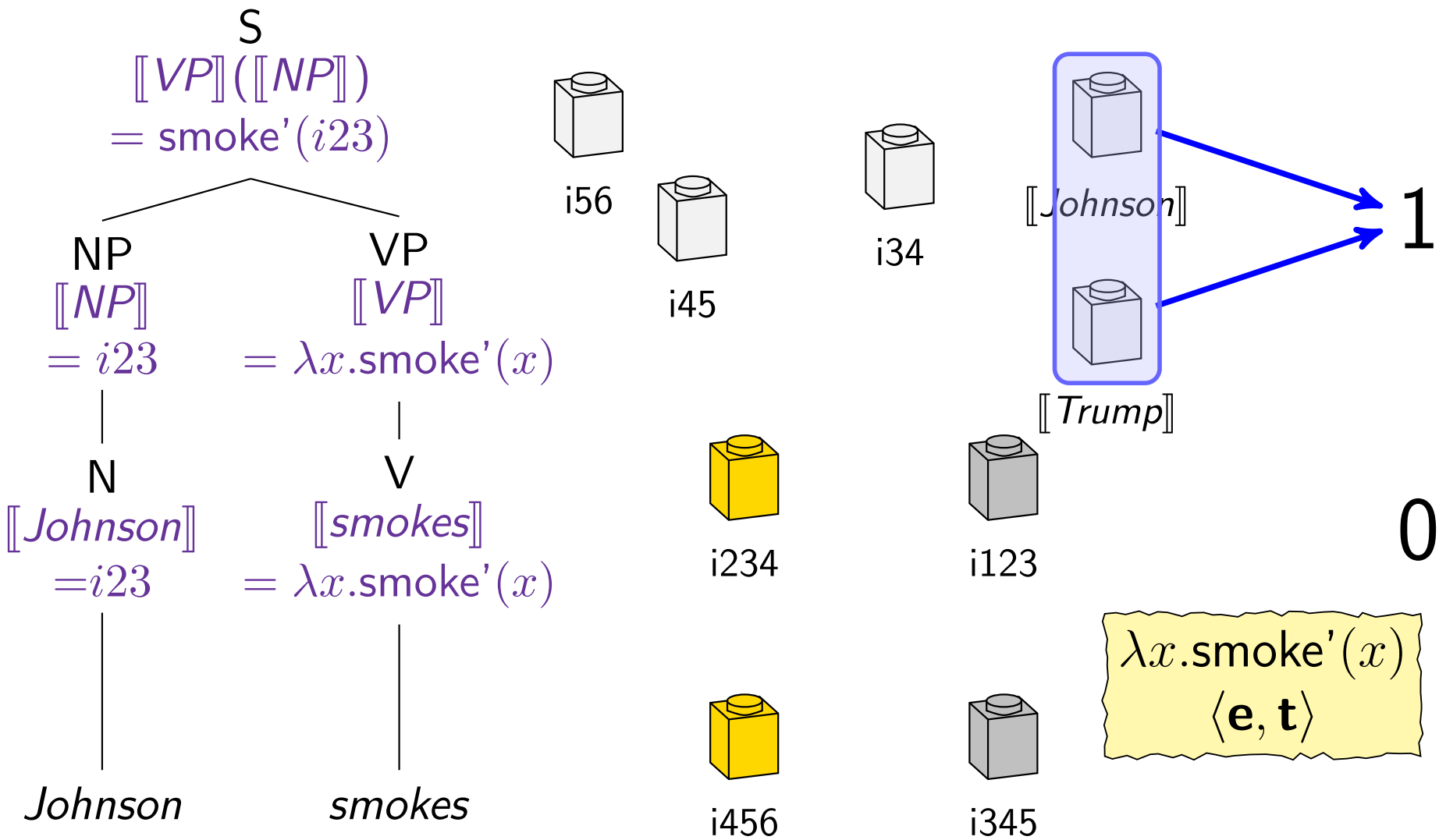


truth value **t**

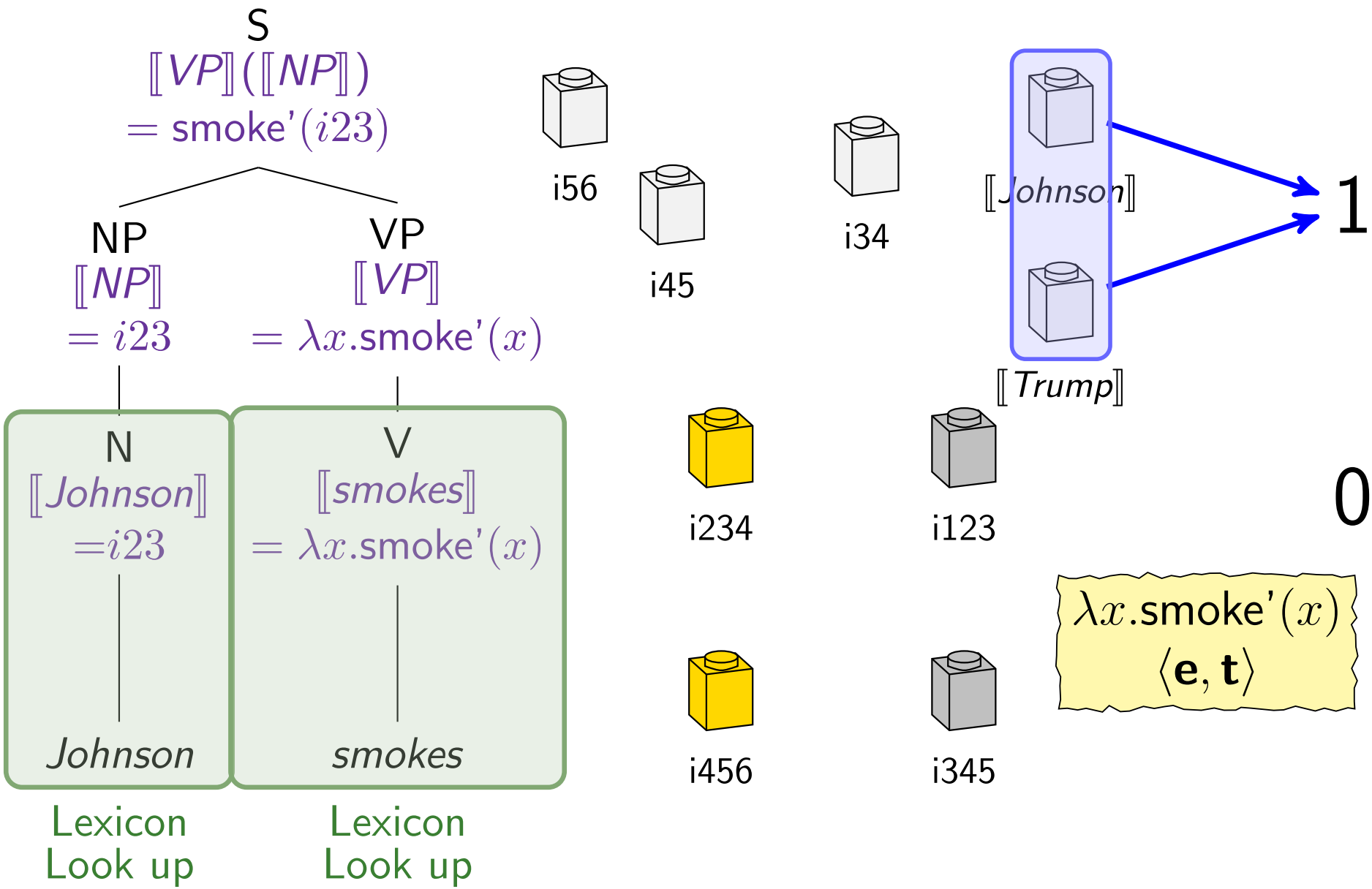


$\lambda x.\text{politician'}(x) : \langle \mathbf{e}, \mathbf{t} \rangle$
$\lambda x.\text{lighter'}(x) : \langle \mathbf{e}, \mathbf{t} \rangle$
$\lambda x.\text{silver'}(x) : \langle \mathbf{e}, \mathbf{t} \rangle$
$\lambda x.\text{golden'}(x) : \langle \mathbf{e}, \mathbf{t} \rangle$

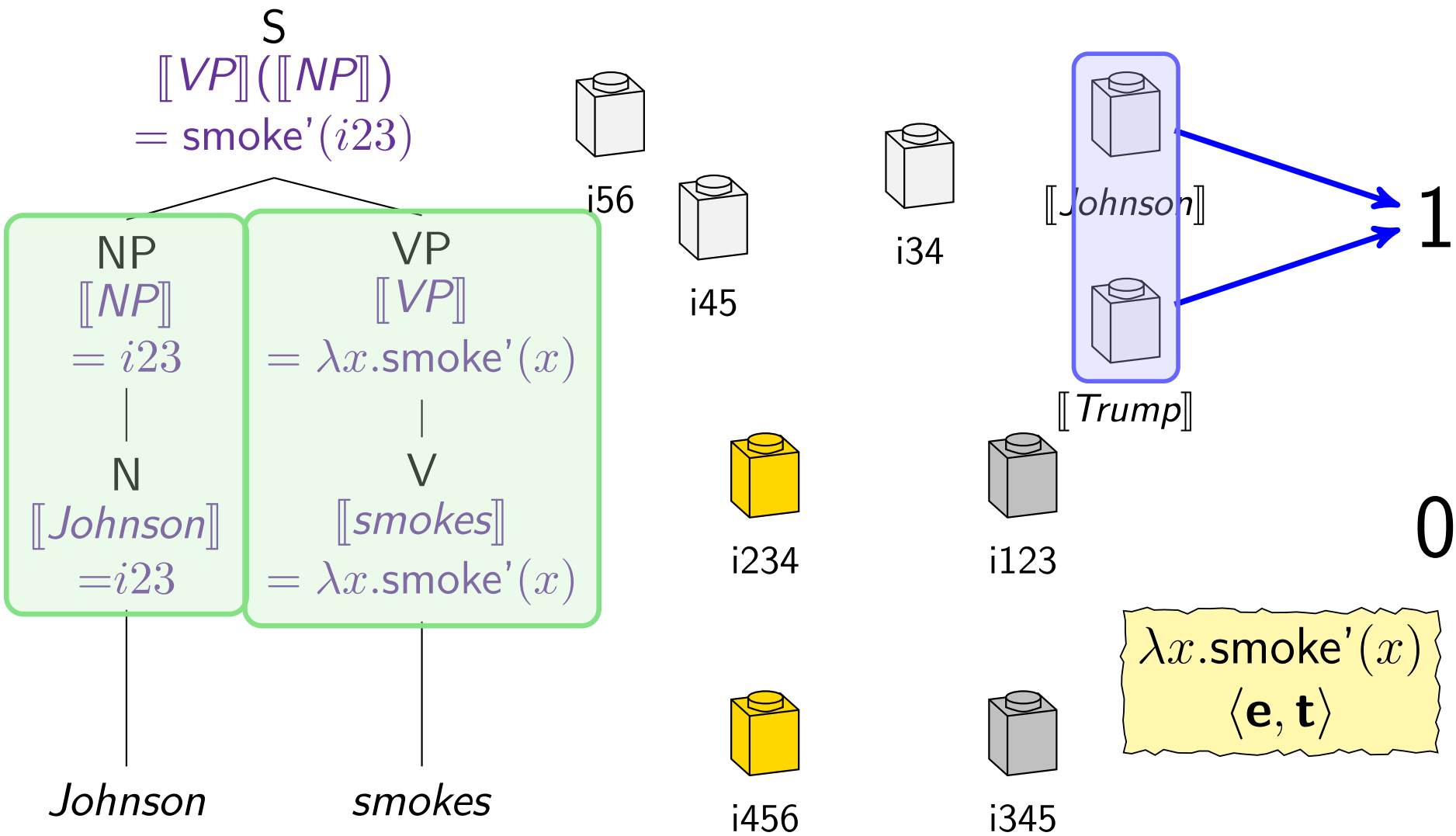
Syntactico-semantic composition



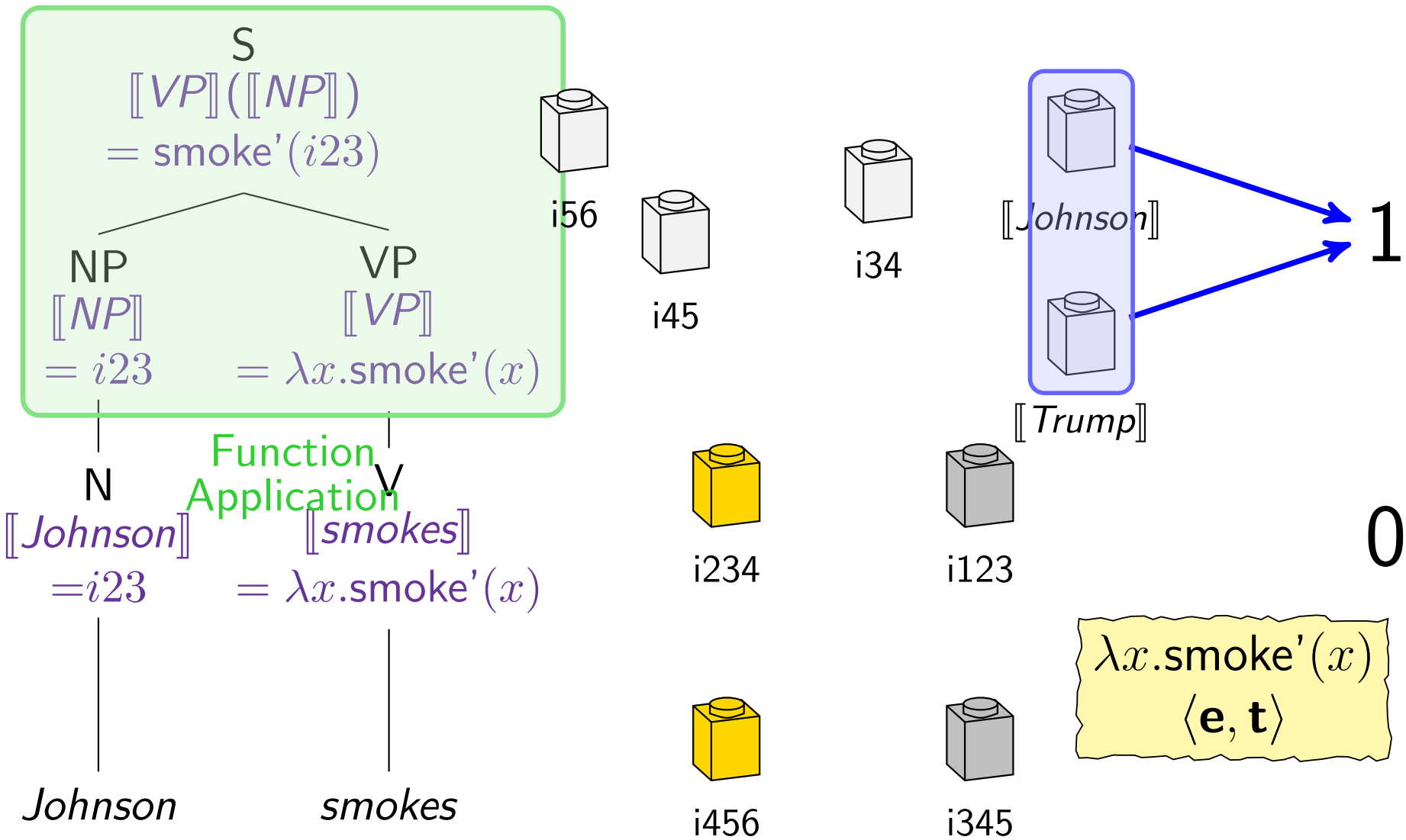
Syntactico-semantic composition



Syntactico-semantic composition



Syntactico-semantic composition

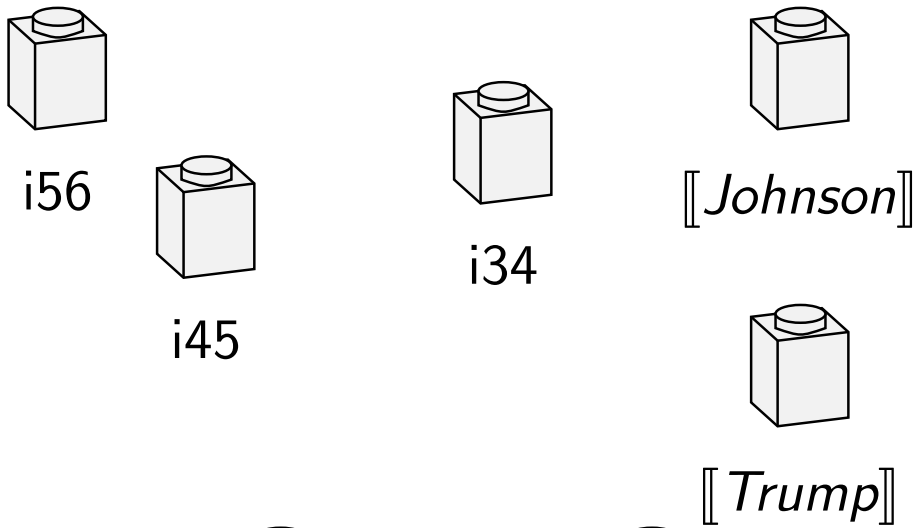


Compositional semantics

- $[\![\textit{Johnson smokes}]\!]$ is not listed in the lexicon.
- But the interpretation of *Johnson smokes* can still be derived from its parts along with a syntactic analysis.
- Finite means make infinite interpretation possible.
- This is exactly the point of compositional semantics
- and note that we have remained precise
- This means we can use this thing we just built as a **meaning representation** of the kind we wanted in Lecture 1.

Transitive verbs

Johnson kissed Trump

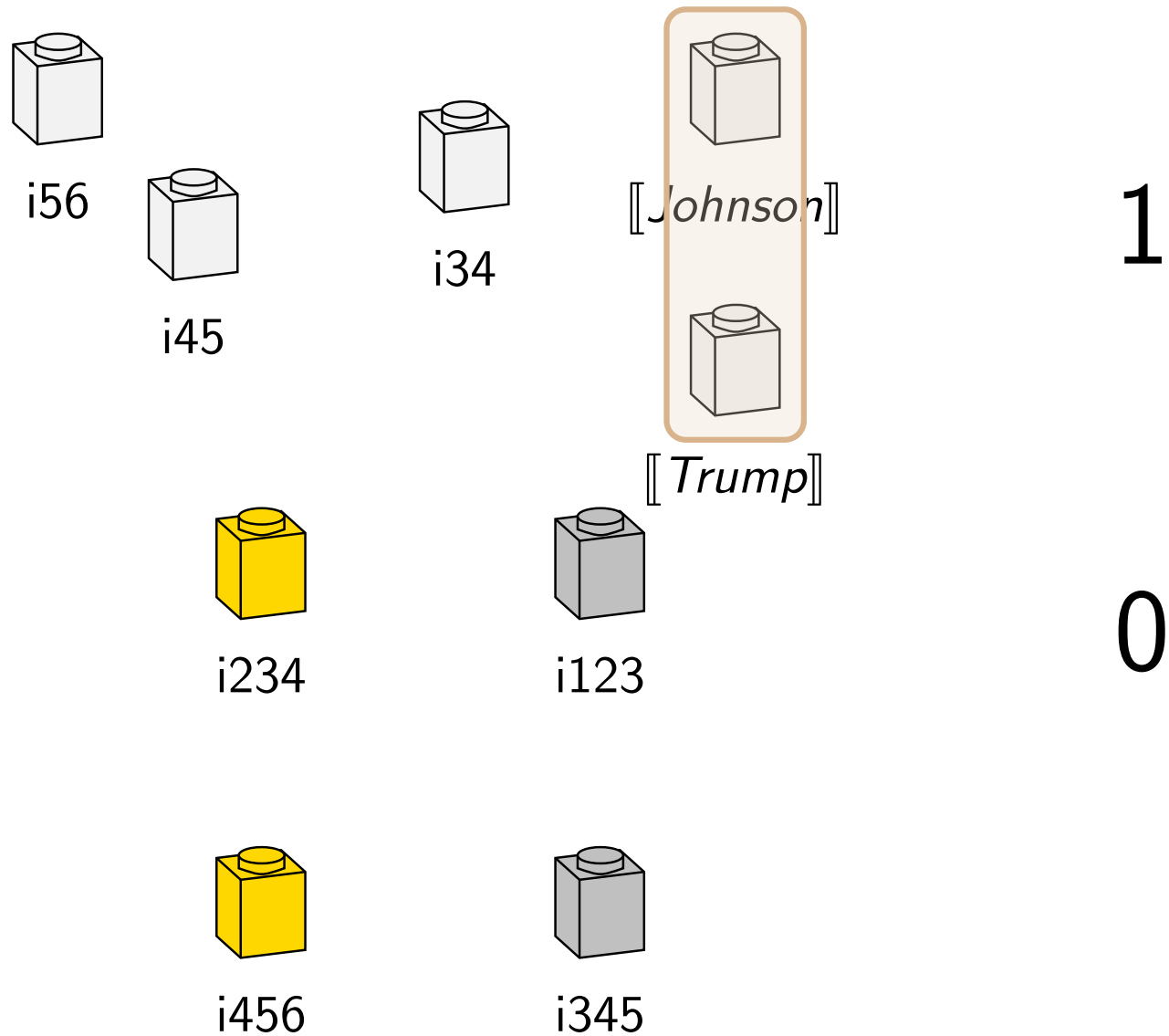


1

0

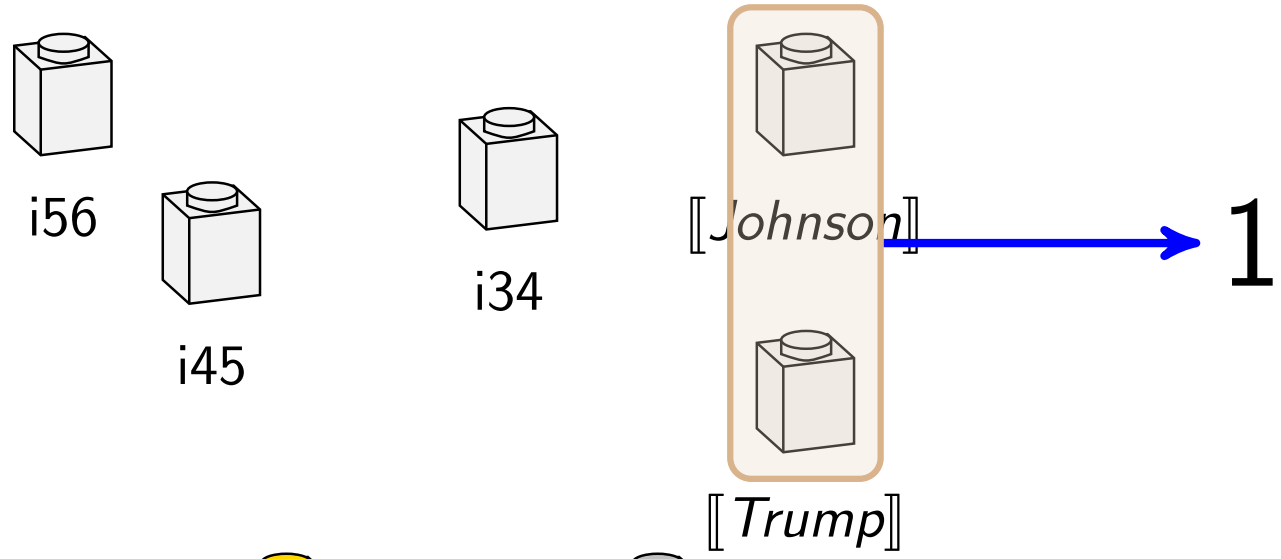
Transitive verbs

Johnson kissed Trump



Transitive verbs

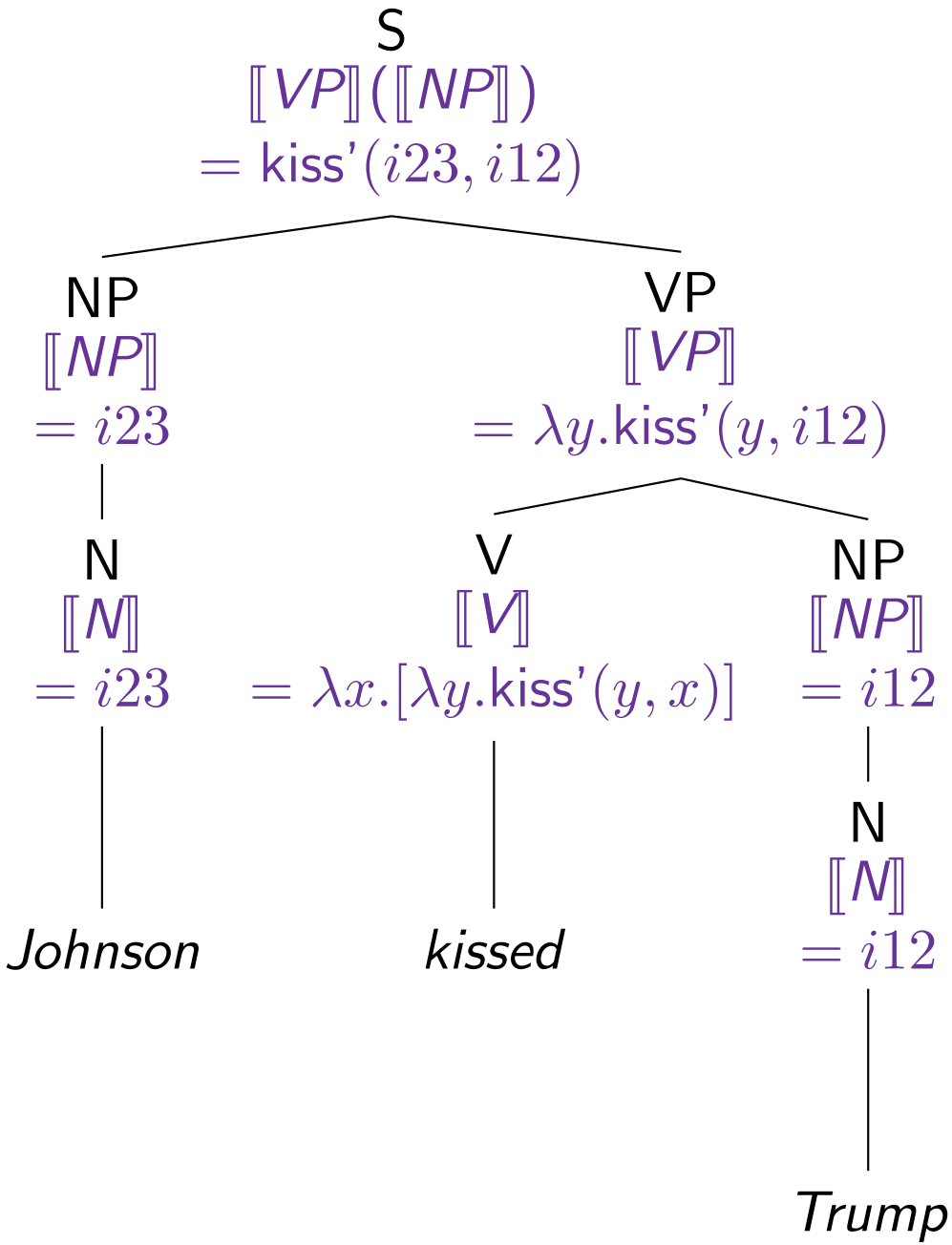
Johnson kissed Trump



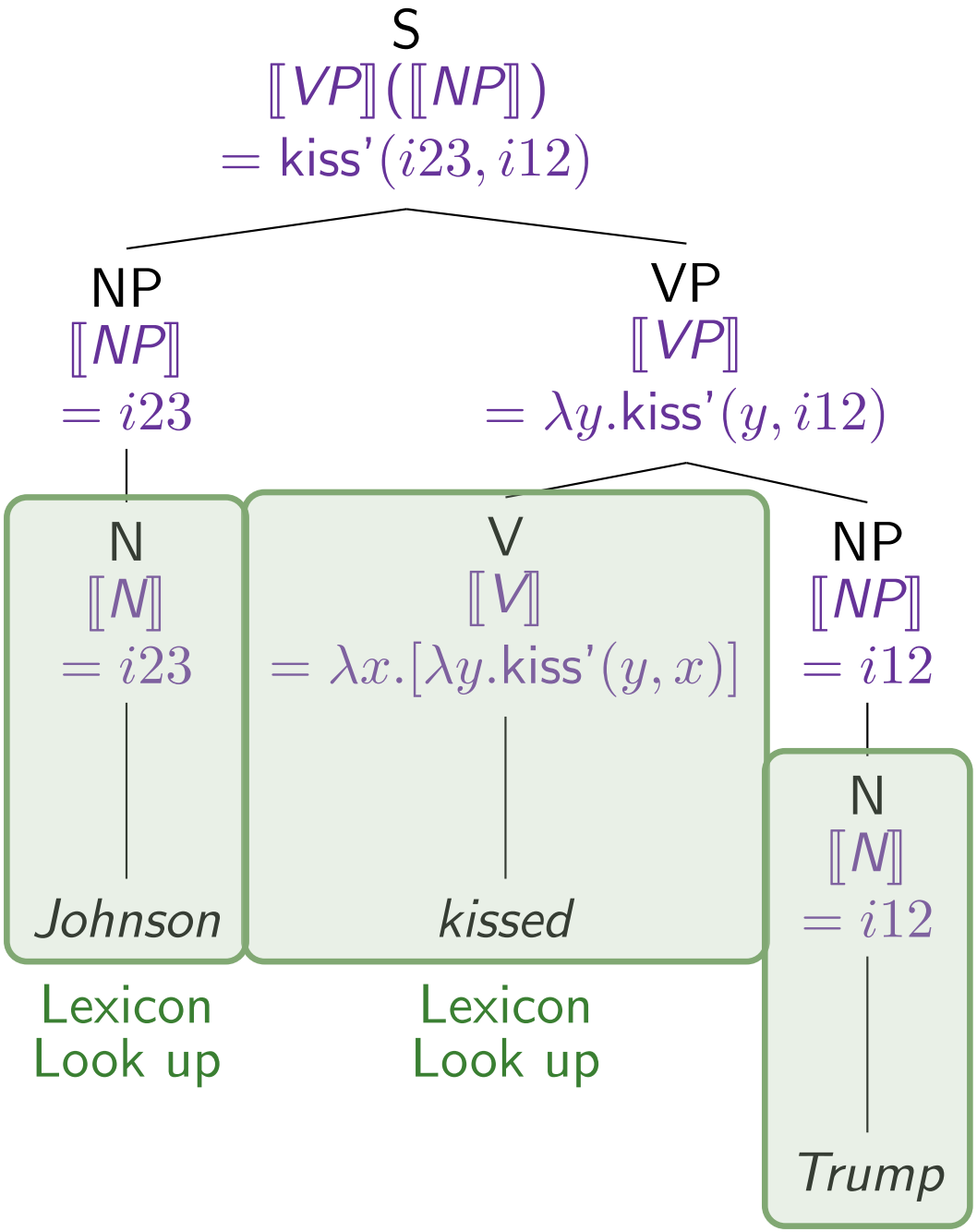
$\lambda x.[\lambda y.\text{kiss'}(y, x)]$

$\langle \mathbf{e}, \langle \mathbf{e}, \mathbf{t} \rangle \rangle$

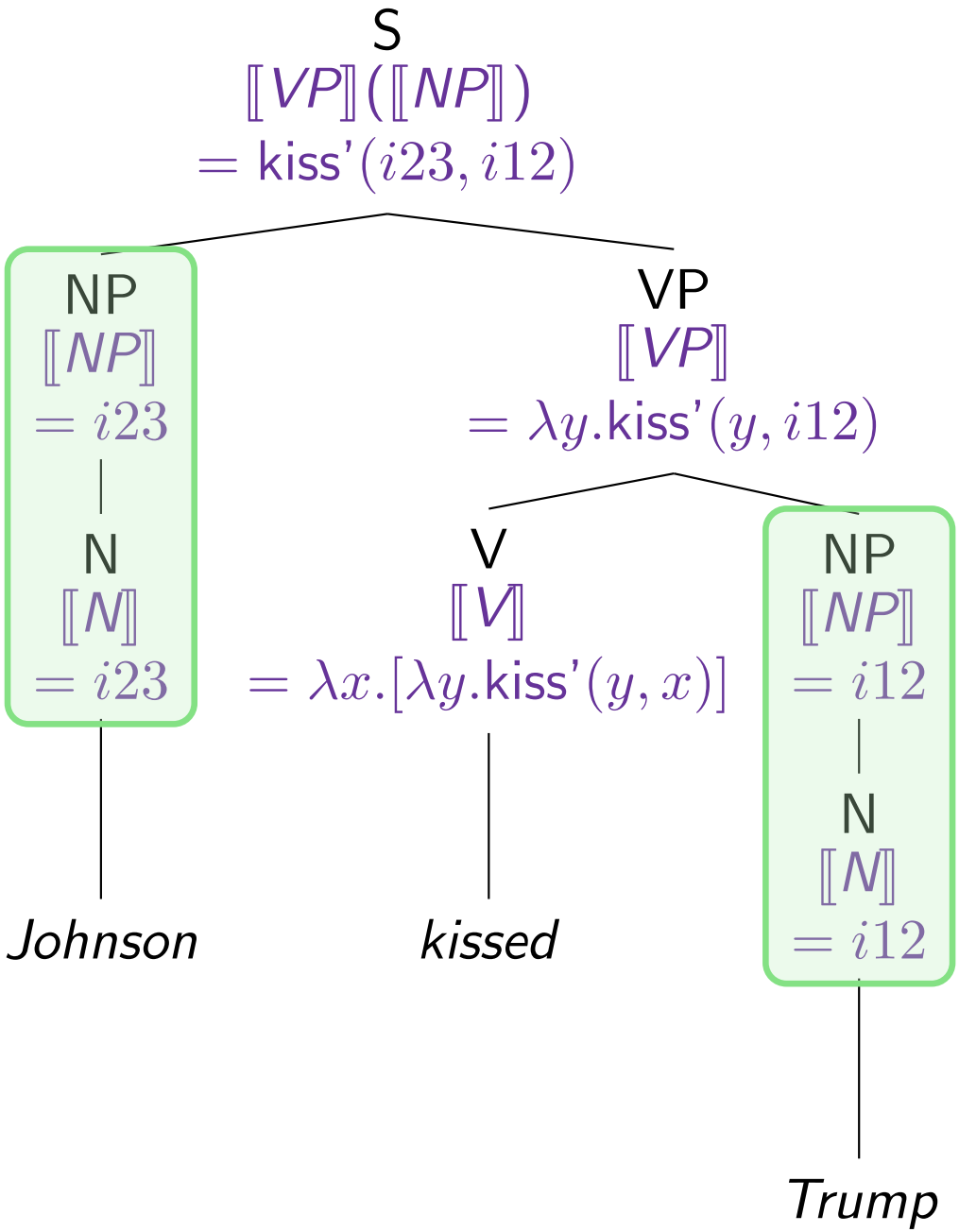
Syntactico-semantic composition



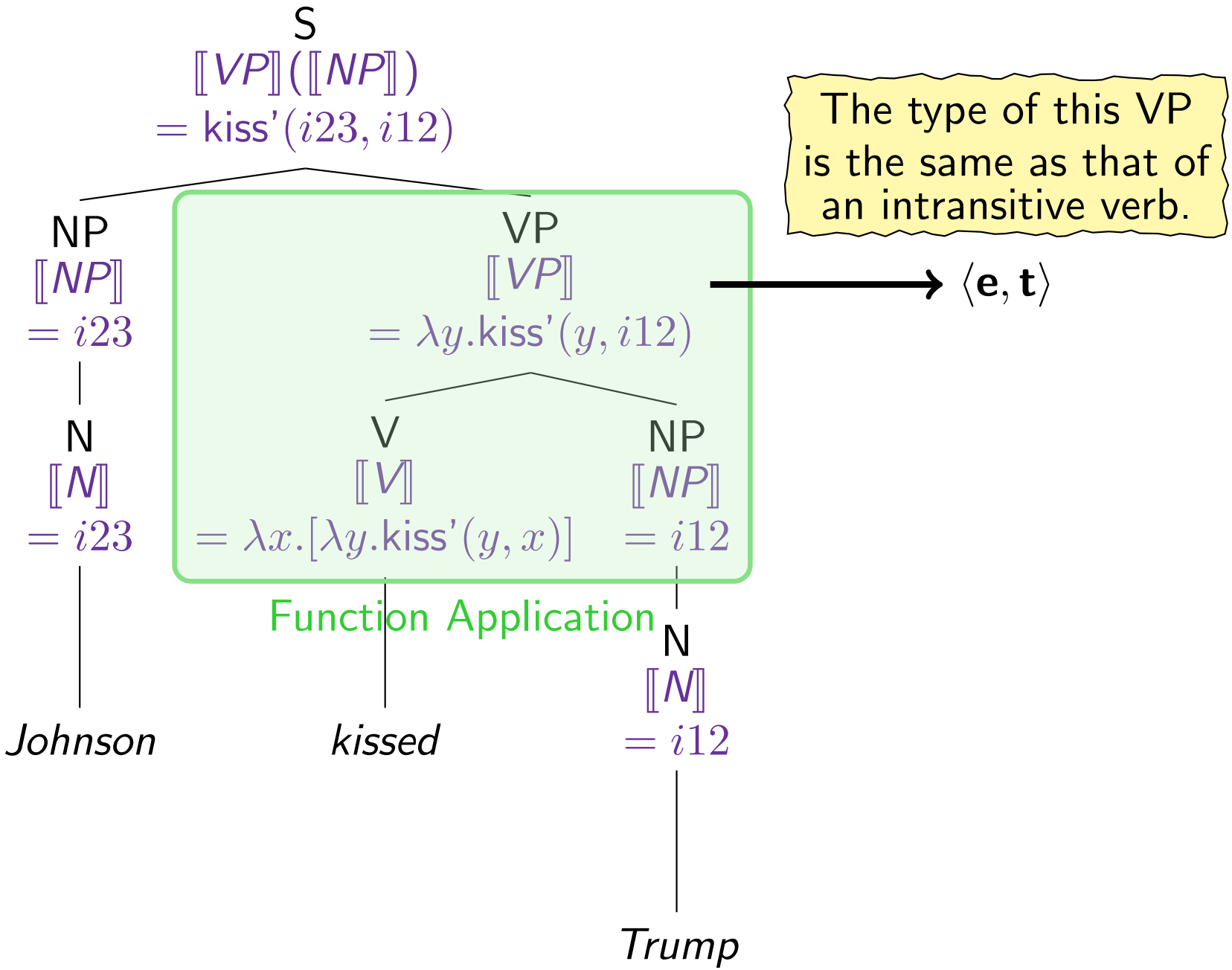
Syntactico-semantic composition



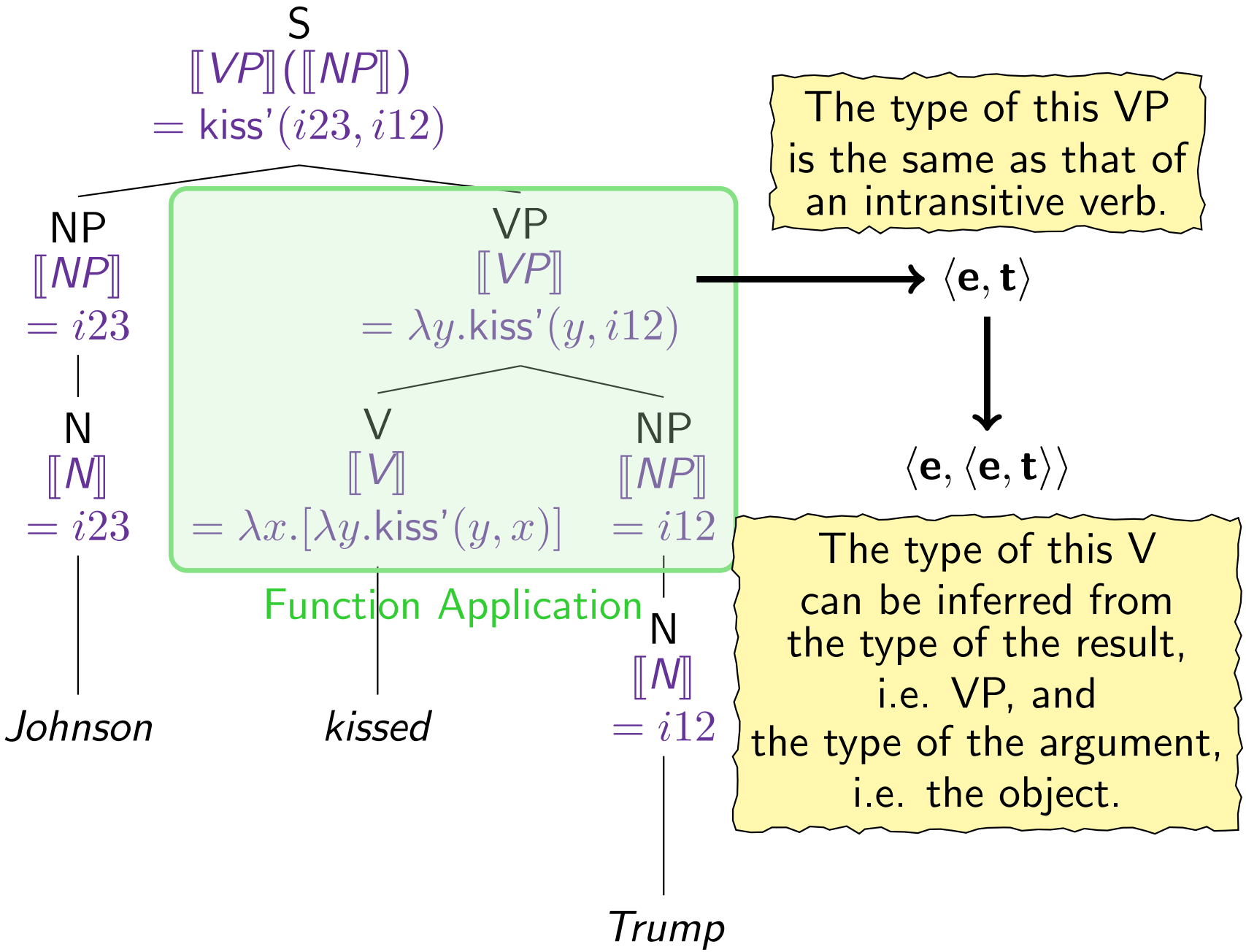
Syntactico-semantic composition



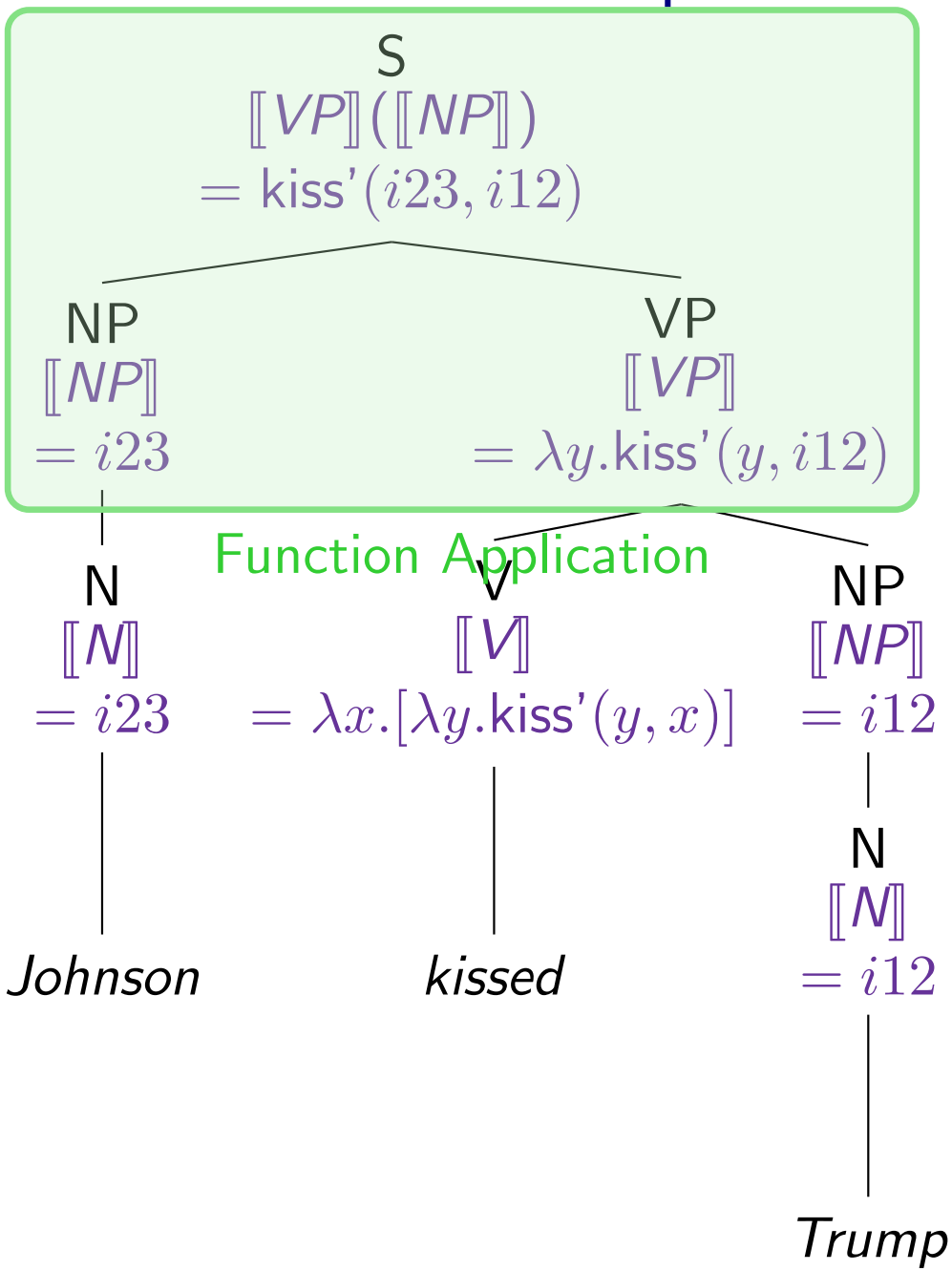
Syntactico-semantic composition



Syntactico-semantic composition



Syntactico-semantic composition



What should we know for a lexical entry?

Example

- *kissed*
- syntactic category: V
- semantic type: $\langle \mathbf{e}, \langle \mathbf{e}, \mathbf{t} \rangle \rangle$
- semantic interpretation: $\lambda x.[\lambda y.\text{kiss'}(y, x)]$

Truth-Conditions

Meanings as truth conditions

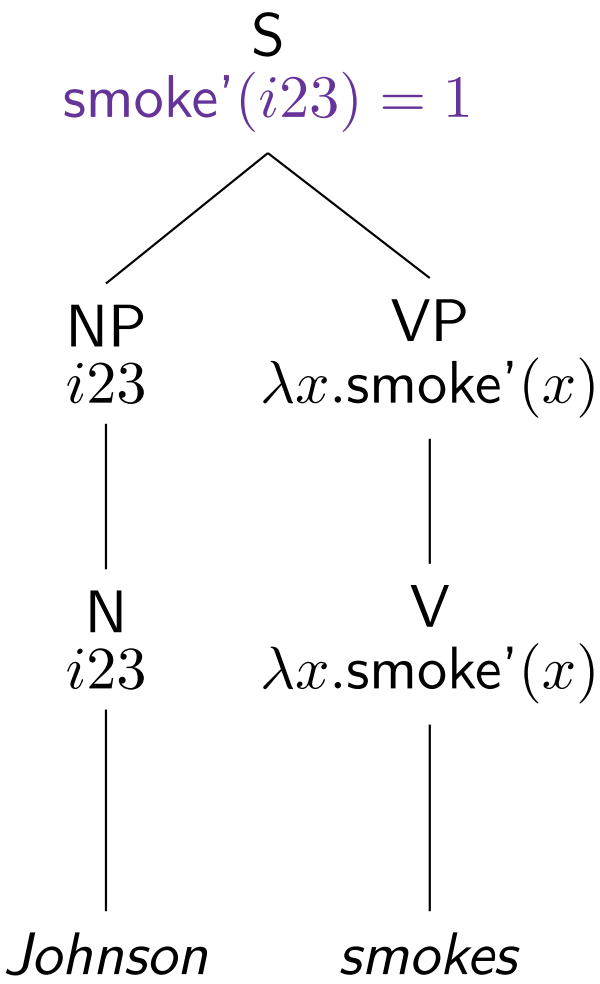
Ludwig Wittgenstein

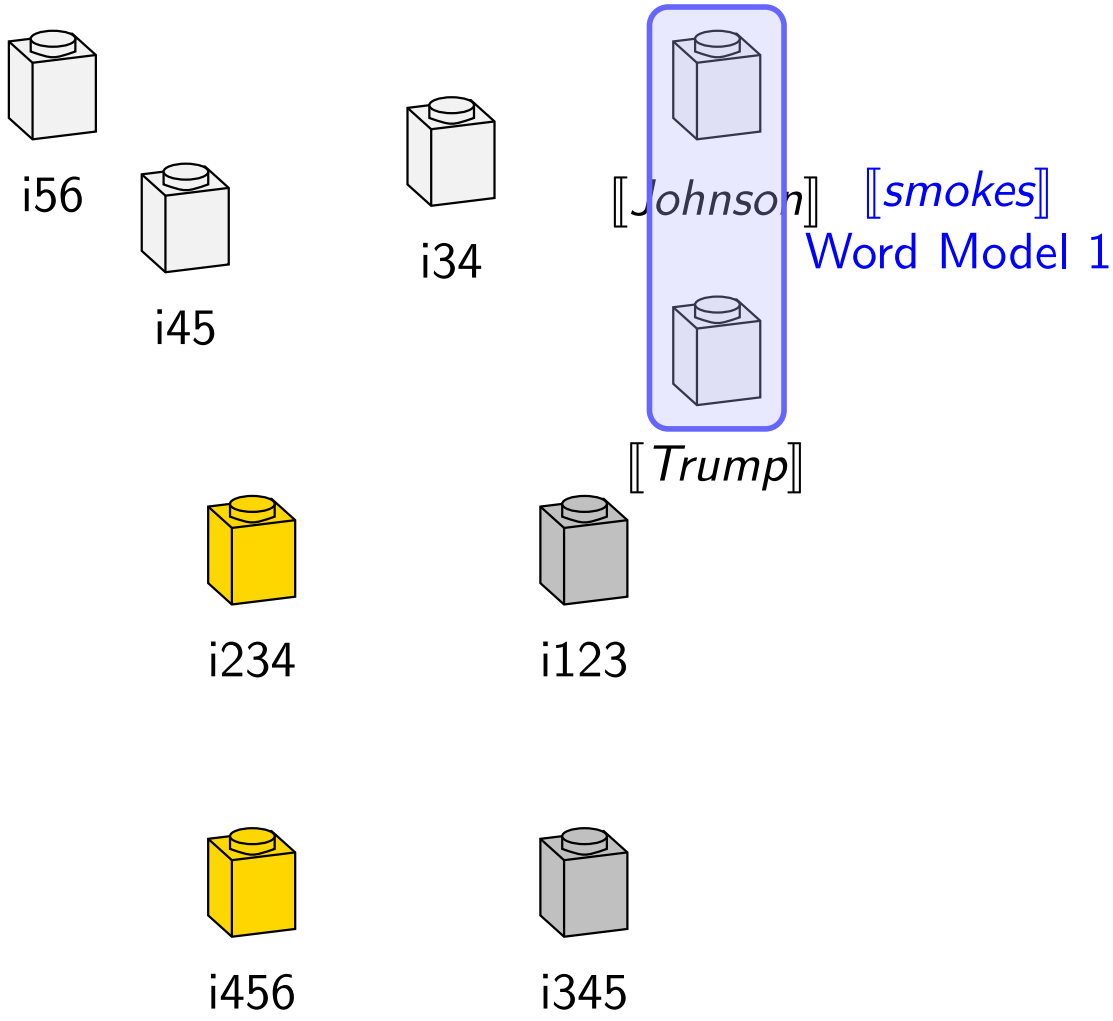
> *To know the meaning of a sentence is to know how the world would have to be for the sentence to be true.*

The meaning of words and sentence parts is their contribution to the truth-conditions of the whole sentence.

The truth-conditional tradition

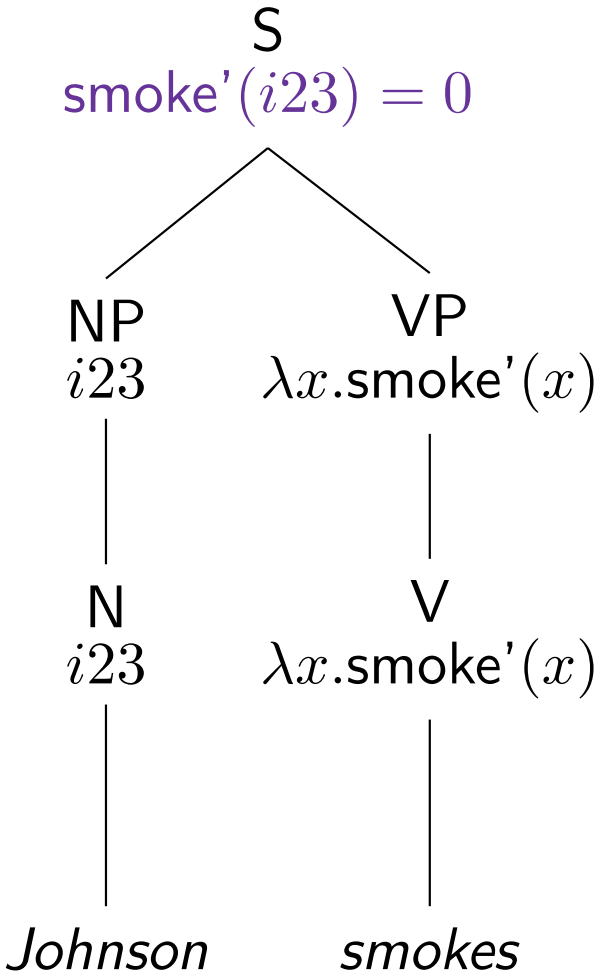
Consider three different word models: Different people smoke

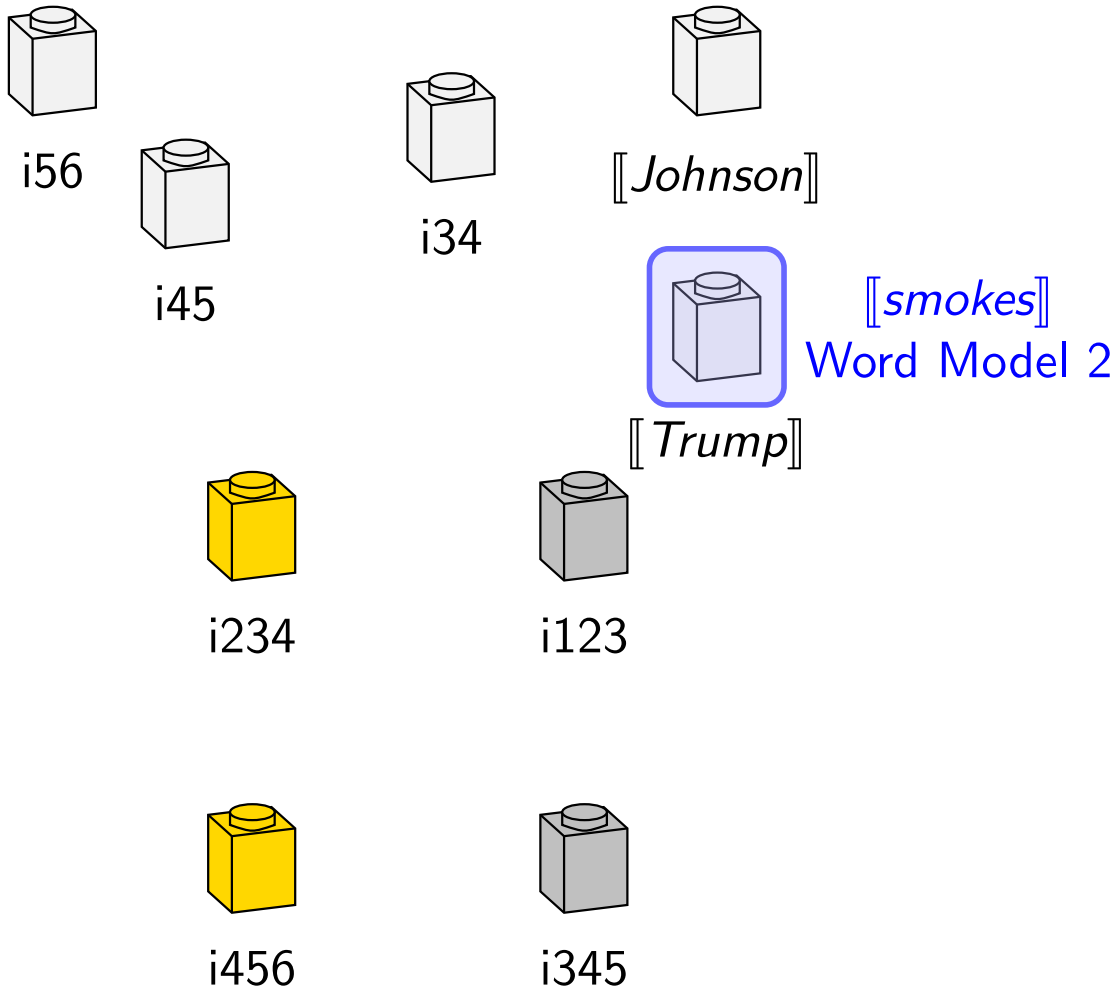




The truth-conditional tradition

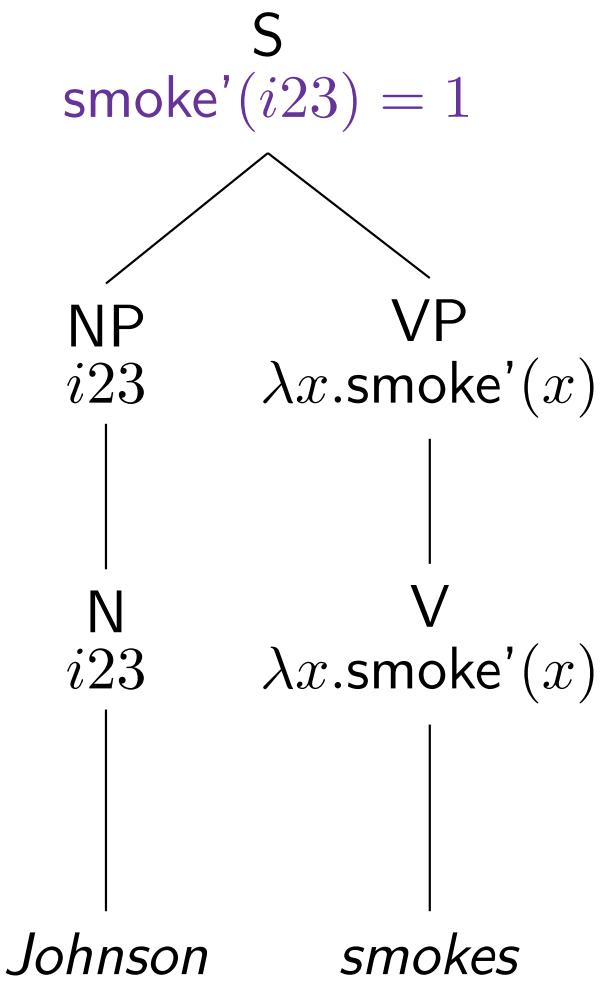
Consider three different word models: Different people smoke

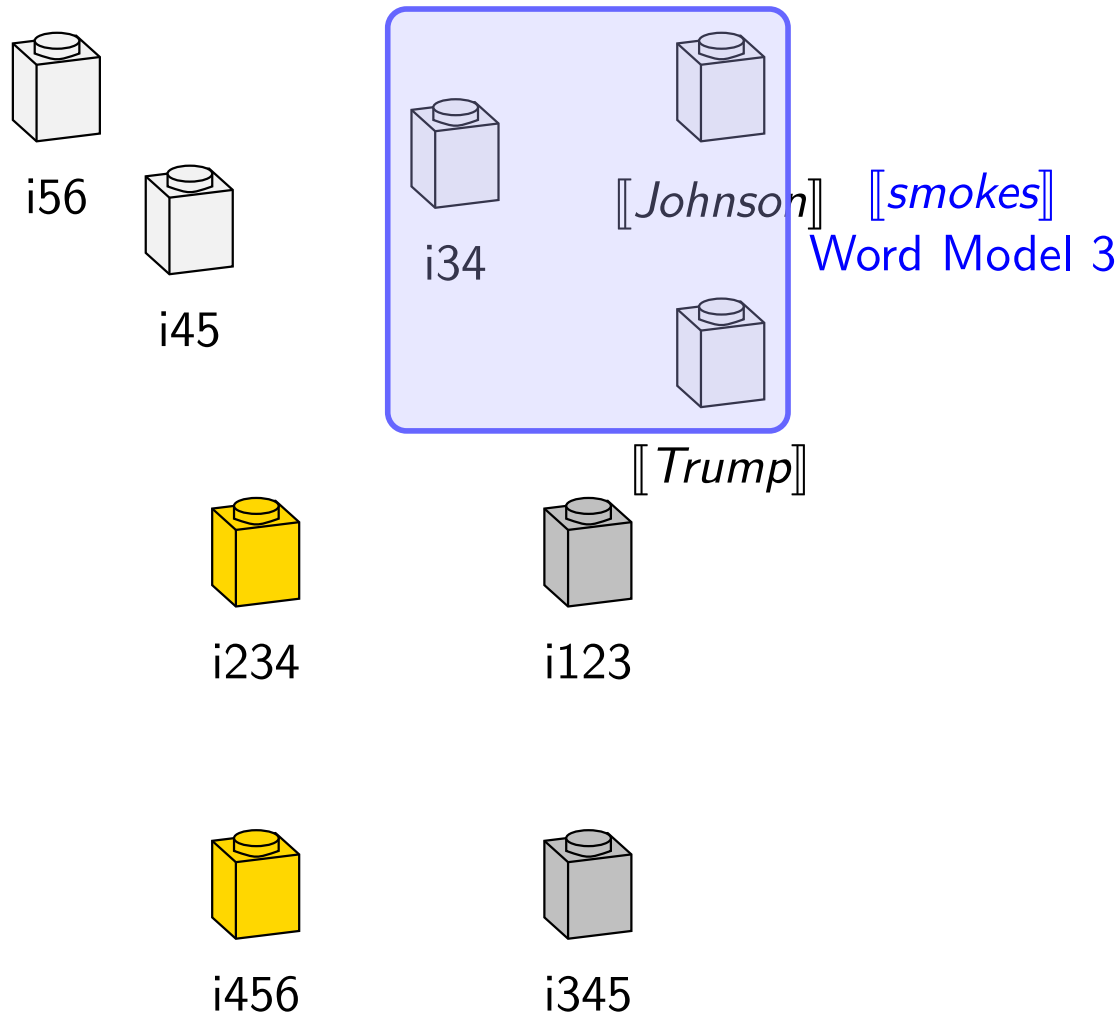




The truth-conditional tradition

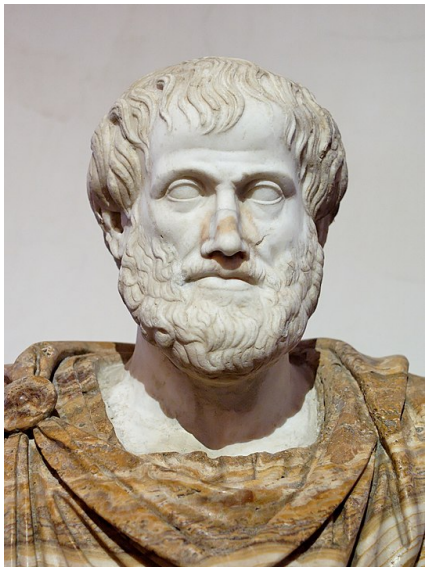
Consider three different word models: Different people smoke





First Order Predicate Logic

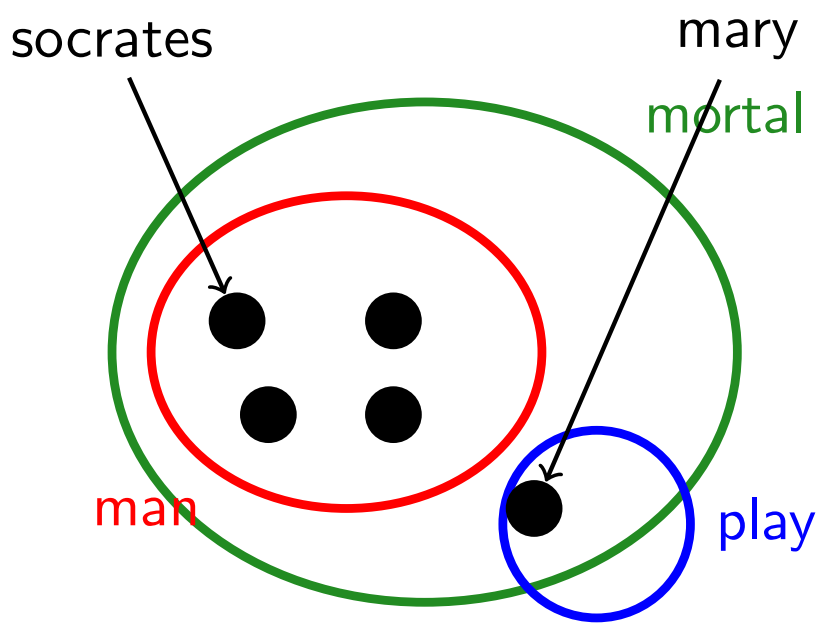
Reasoning, Syllogism=Syn- + logos



- All men are mortal.
- Socrates is a man.
- Therefore, Socrates is mortal.

- all': $\langle\langle \mathbf{e}, \mathbf{t}\rangle, \langle \mathbf{e}, \mathbf{t}\rangle\rangle$ second-order
- man': $\langle \mathbf{e}, \mathbf{t}\rangle$
- mortal': $\langle \mathbf{e}, \mathbf{t}\rangle$

M1



FOPL syntax, Alphabet

- Quantifier symbols: $\forall$, $\exists$
- Logical connectives: $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\neg$
- Infinite set of variables, e.g. x, y, z
- Equality symbol $=$
- Truth constants $\top$ and $\bot$
- Infinite set of n-ary predicate symbols, e.g. P, Q
- Infinite set of n-ary function symbols, e.g. f, g

FOPL syntax, Term

Terms in FOPL are defined as:

- Any variable symbol is a term
- If f is an n-ary function symbol, and t_1, ..., t_n are terms, then $f(t_1, \ldots, t_n)$ is a term.

FOPL syntax, Formula

Formulae in FOPL are defined as follows:

- Predicate symbols. If P is an n-ary predicate symbol and t_1,. . . , t_n are terms then $P(t_1, ..., t_n)$ is a formula.
- Equality. If the equality symbol is considered part of the logic, and t_1 and t_2 are terms, then $t1 = t2$ is a formula.
- Negation. If φ is a formula, then $\neg\varphi$ is a formula.
- Binary connectives. If φ and ψ are formulas, then $\varphi \rightarrow \psi$ is a formula. Similar rules apply to other binary logical connectives.
- Quantifiers. If φ is a formula and x is a variable, then $\forall x\varphi$ and $\exists x\varphi$ are formulas.

FOPL syntax, notational conventions

- $\neg$ is evaluated first
- $\wedge$ and $\vee$ are evaluated next
- Quantifiers are evaluated next
- $\rightarrow$ is evaluated last.
- (and) can be used to explictly indicate combination orders.

FOPL, Examples

Example: a well-formed formula

$$\forall x \forall y (P(f(x)) \rightarrow \neg (P(x) \rightarrow Q(f(y), x, z)))$$

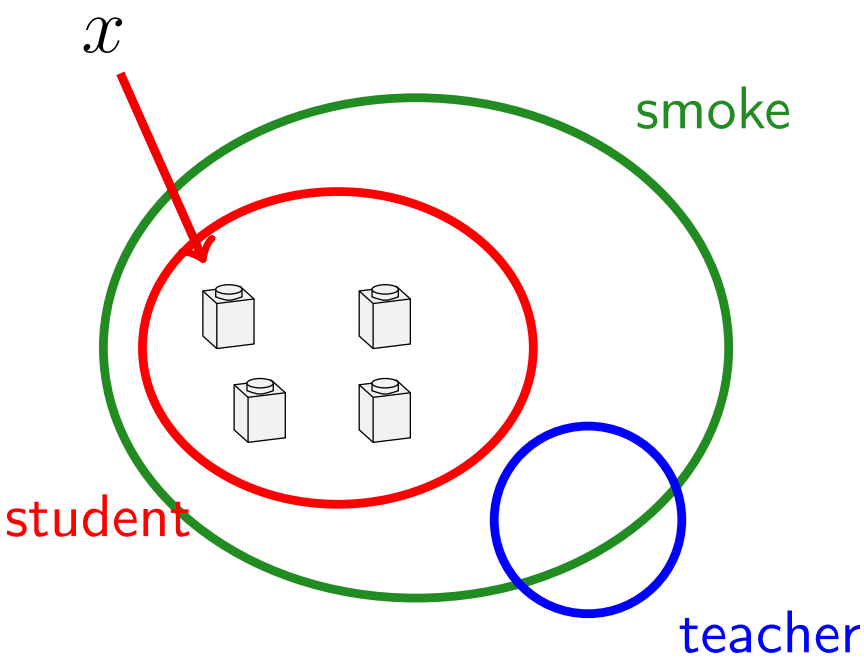
is a well-formed formula, if f is a unary function symbol, P a unary predicate symbol, and Q a ternary predicate symbol.

Example: a string of symbols from the alphabet that is not a formula

$$\forall x \, x \rightarrow$$

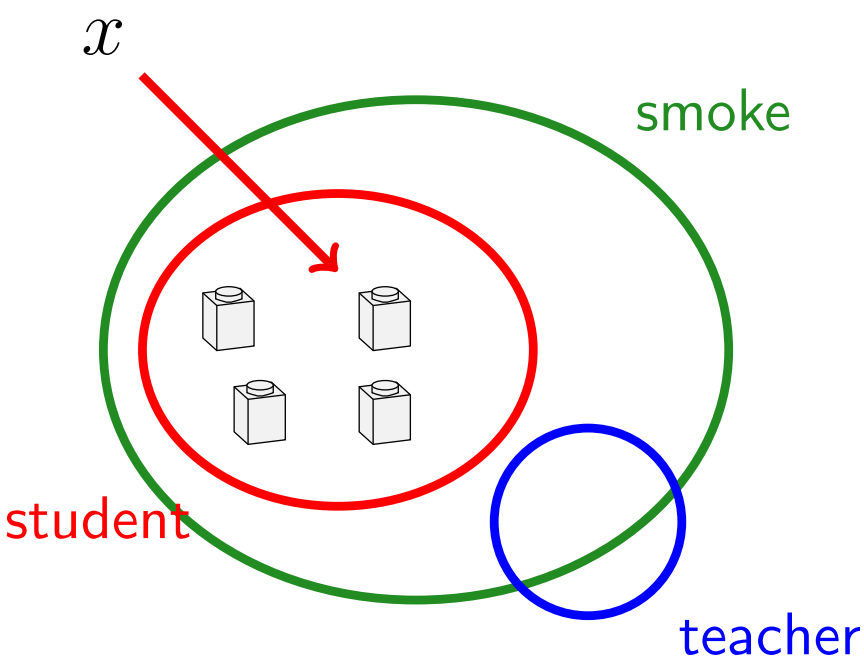
Universal quantifier

- What is $[\![$*every student smokes*$]\!]$? $\forall x(\text{student'}(x) \rightarrow \text{smoke'}(x))$
- What is $[\![$*some students smoke*$]\!]$? $\exists x(\text{student'}(x) \wedge \text{smoke'}(x))$



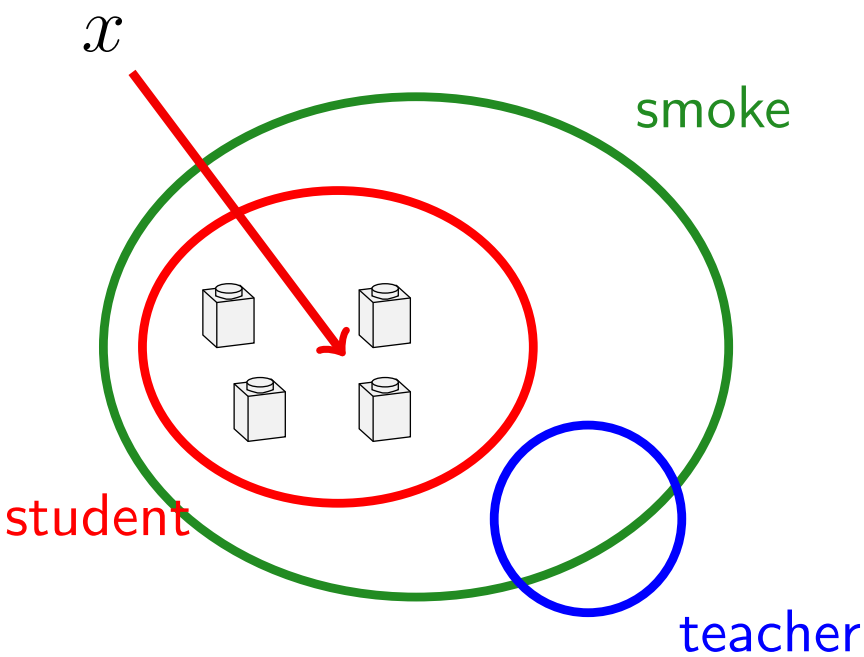
Universal quantifier

- What is $[\![\textit{every student smokes}]\!]$? $\forall x(\text{student'}(x) \rightarrow \text{smoke'}(x))$
- What is $[\![\textit{some students smoke}]\!]$? $\exists x(\text{student'}(x) \wedge \text{smoke'}(x))$



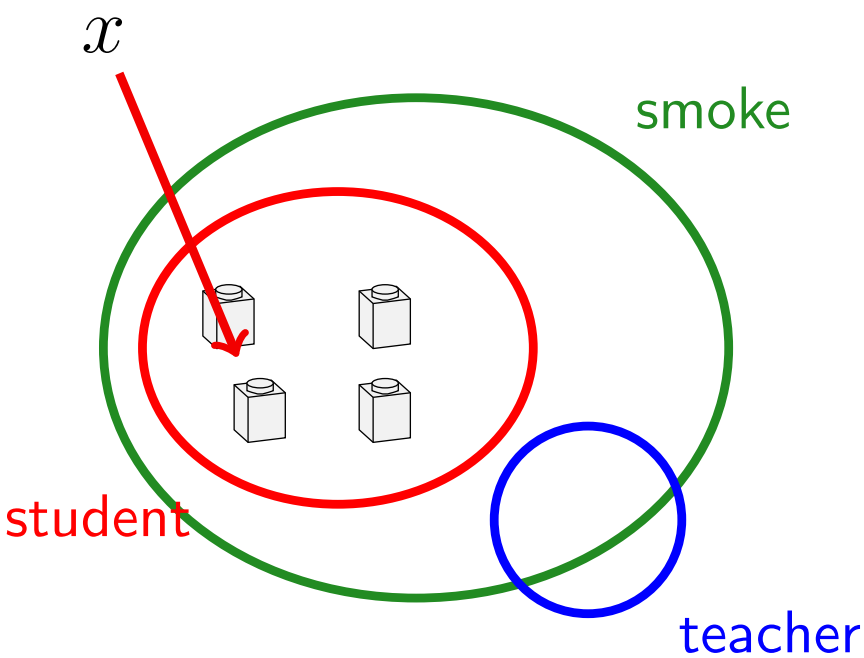
Universal quantifier

- What is $[\![\textit{every student smokes}]\!]$? $\forall x(\text{student'}(x) \rightarrow \text{smoke'}(x))$
- What is $[\![\textit{some students smoke}]\!]$? $\exists x(\text{student'}(x) \wedge \text{smoke'}(x))$



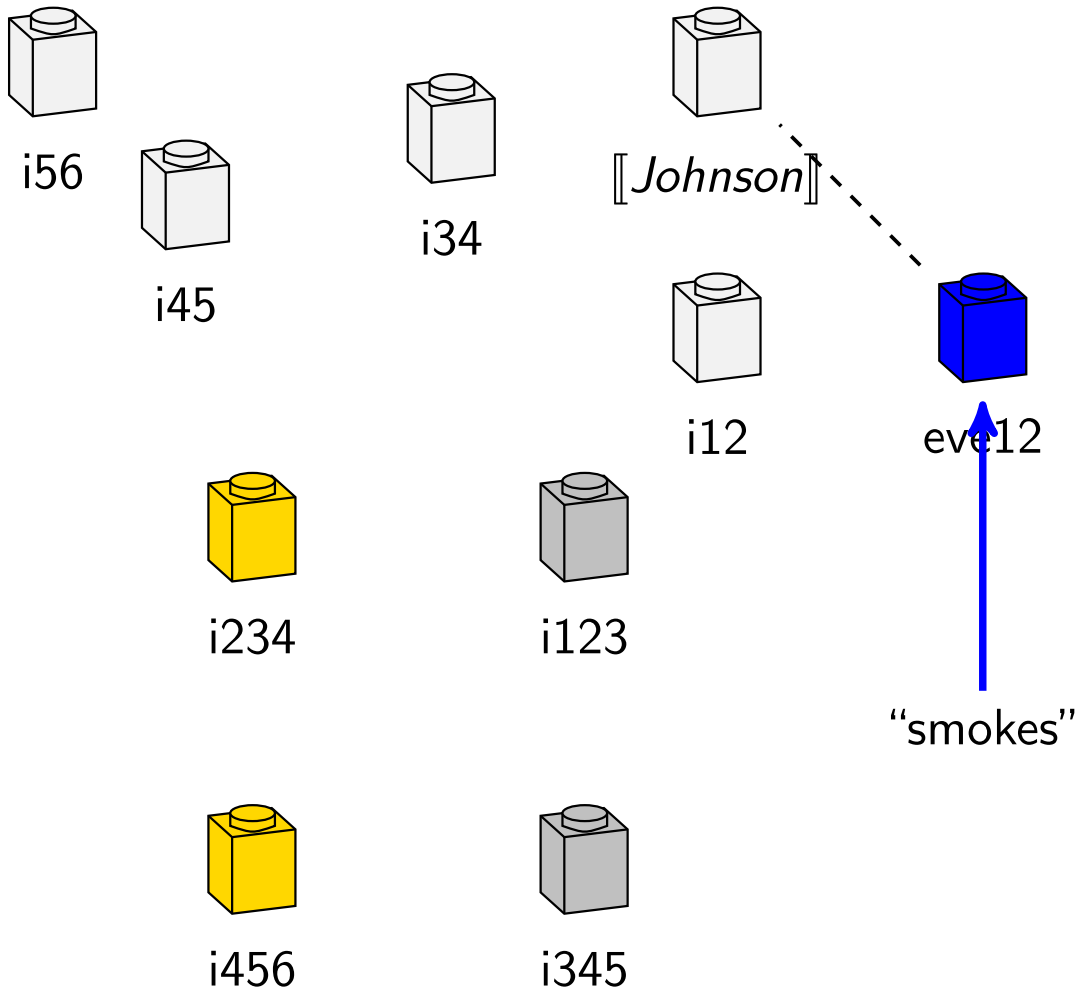
Universal quantifier

- What is $[\![$*every student smokes*$]\!]$? $\forall x(\text{student'}(x) \rightarrow \text{smoke'}(x))$
- What is $[\![$*some students smoke*$]\!]$? $\exists x(\text{student'}(x) \wedge \text{smoke'}(x))$

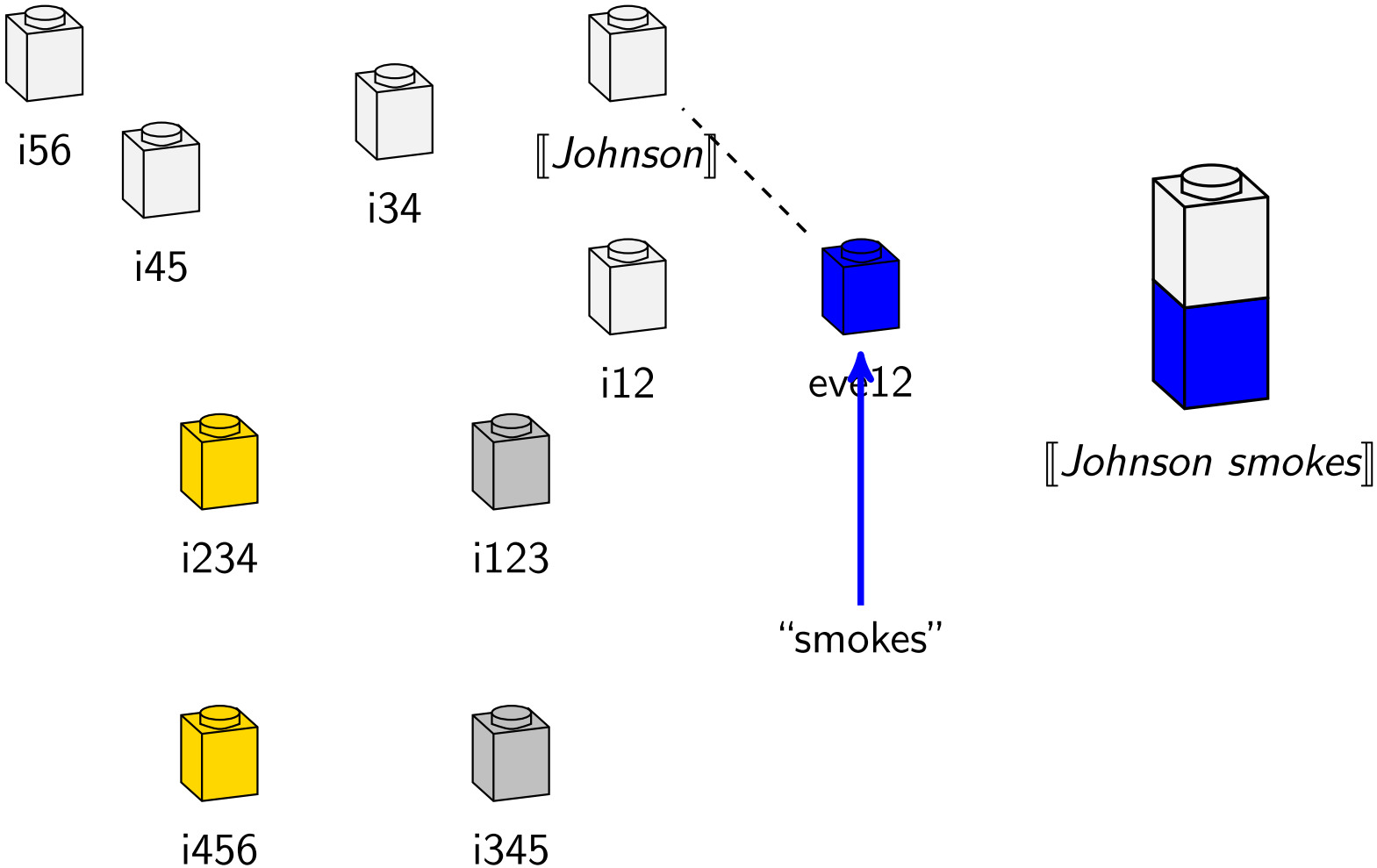


Davidsonian Semantics

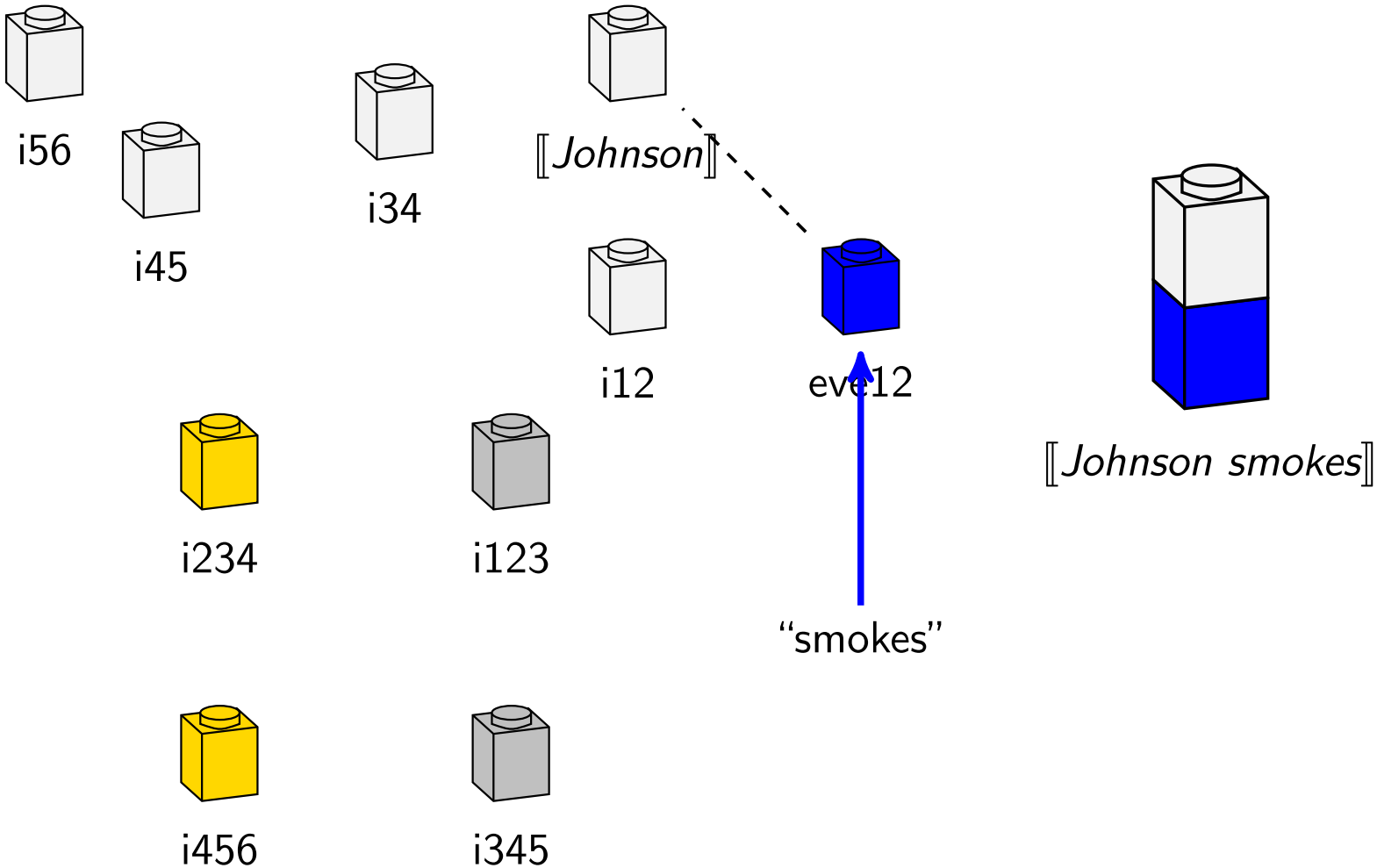
Davidsonian semantics: Adding event variables



Davidsonian semantics: Adding event variables

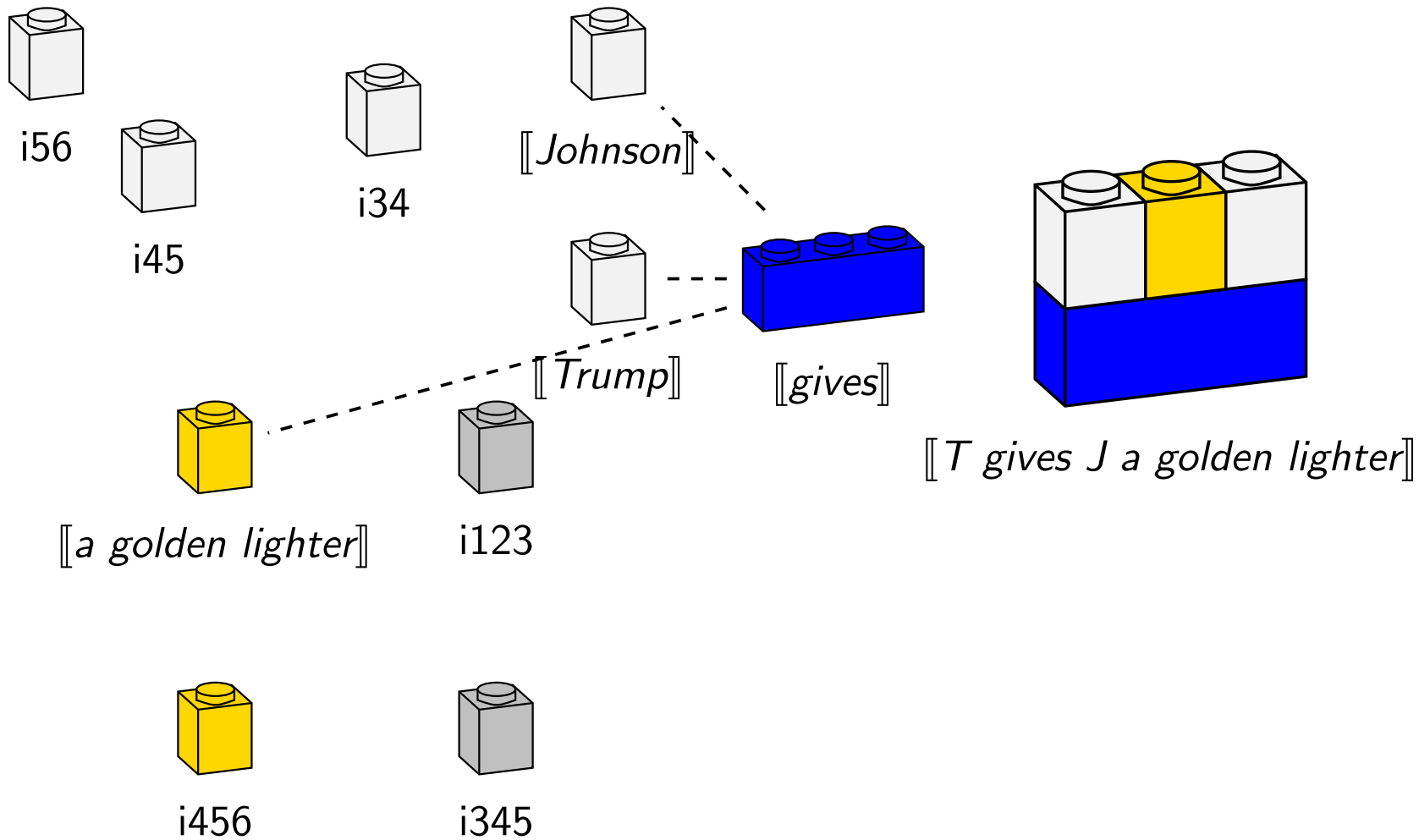


Davidsonian semantics: Adding event variables

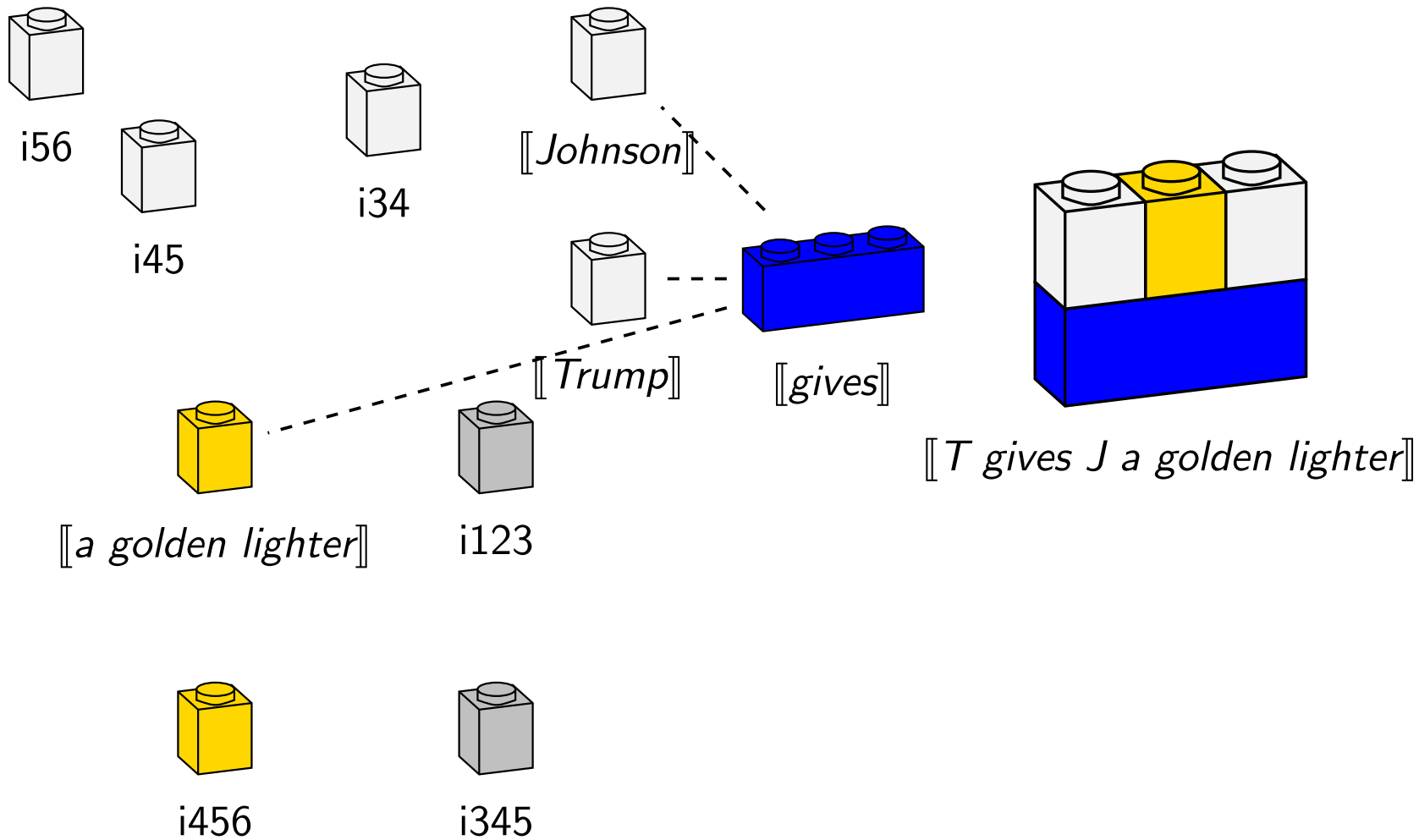


What is the type of ⟦*gives*⟧? **e** — individual.

Ditransitive verb

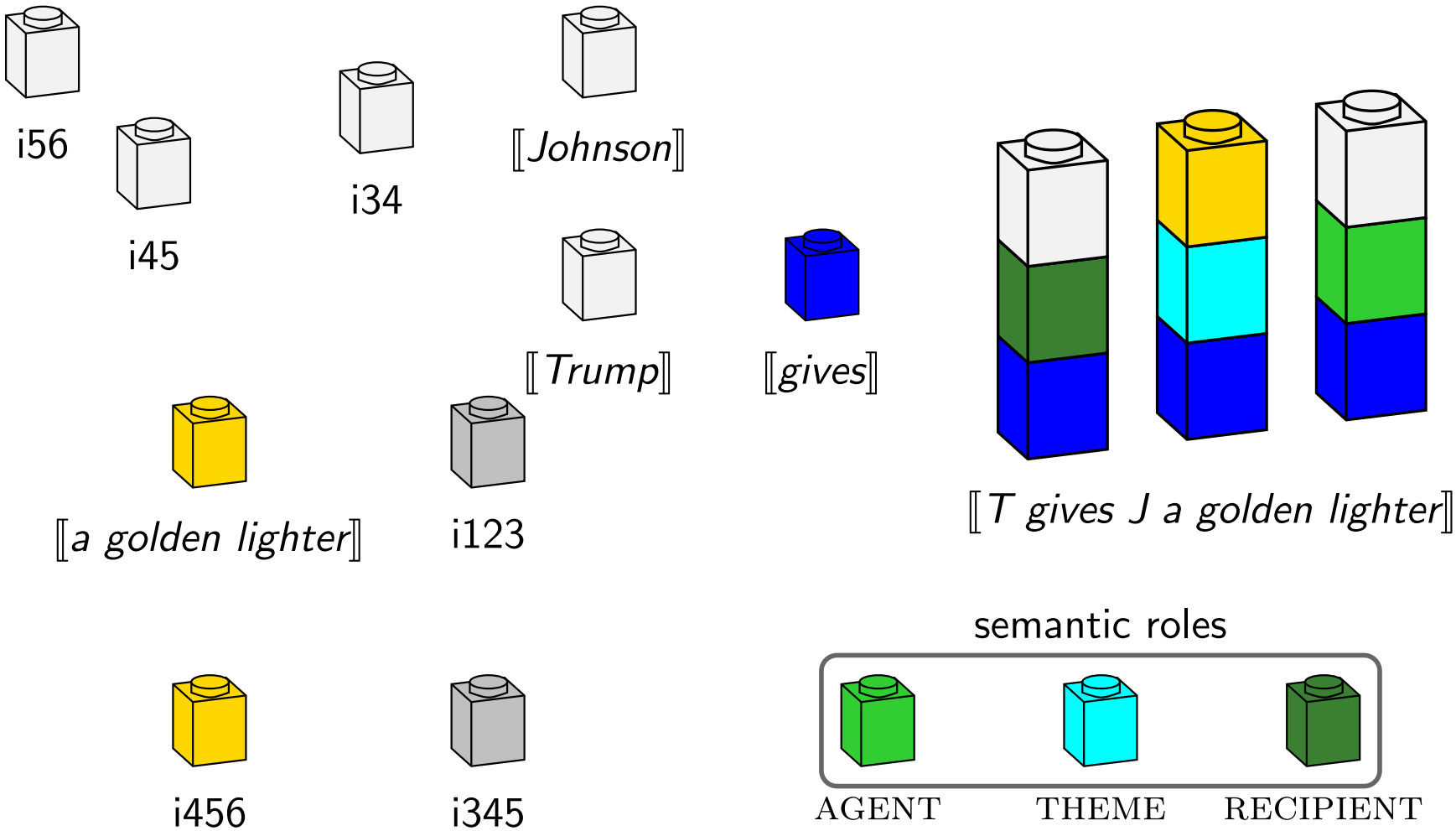


Ditransitive verb

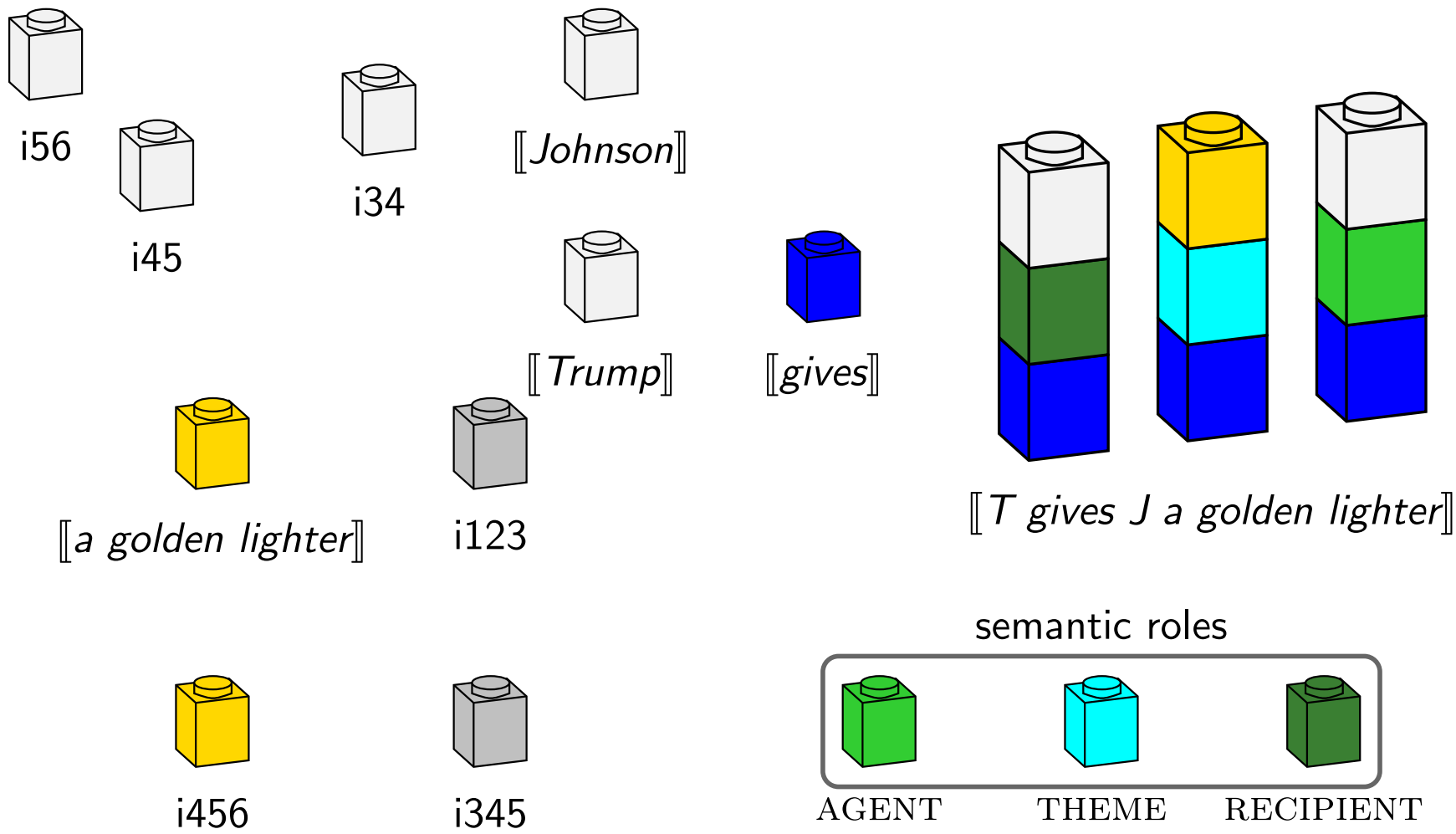


What is the type of ⟦*gives*⟧? **e** — individual.

Neo-Davidsonian semantics: Further decomposition



Neo-Davidsonian semantics: Further decomposition



Further decomposition of the event structure

Lexicalised vs unlexicalised

Before Davidson

- $[\![\textit{gives}]\!]([\![\textit{Trump}]\!], [\![\textit{Johnson}]\!], [\![\textit{a golden lighter}]\!])$
- $\lambda x.[\lambda y.[\lambda z.\text{give'}(z, x, y)]]$ $\langle \mathbf{e}, \langle \mathbf{e}, \langle \mathbf{e}, \mathbf{t} \rangle \rangle \rangle$
- Lexicalised: the lexical entry contains rich information of arguments.

Davidsonian

- $[\![\textit{gives}]\!](e, [\![\textit{Trump}]\!], [\![\textit{Johnson}]\!], [\![\textit{a golden lighter}]\!])$
- Lexicalised
- Now it is easier to handle Mandarin verbal classifiers.

Neo-Davidsonian

- $[\![\textit{gives}]\!](e) \wedge \textsc{agent}(e, [\![\textit{Trump}]\!]) \wedge \textsc{recipient}(e, [\![\textit{Johnson}]\!]) \wedge \textsc{theme}(e, [\![\textit{a golden lighter}]\!])$
- Modularisation of information
- Unlexicalised: the lexical entry doesn't need to know argument structure.

Truth of these statements in our world model?

In the world where *Trump gave Johnson a golden lighter* is true, which one of the following is true?

- Johnson gave Trump a lighter
- Trump gave Johnson a silver lighter
- Johnson was given a lighter

Truth of these statements in our world model?

Remember the world where Trump gave Johnson a golden lighter? Are the following statements true in that world?

1. Johnson gave Trump a lighter
2. Trump gave Johnson a silver lighter
3. Johnson was given a lighter

1. Johnson gave Trump a lighter.

Truth of these statements in our world model?

Remember the world where Trump gave Johnson a golden lighter? Are the following statements true in that world?

1. Johnson gave Trump a lighter
2. Trump gave Johnson a silver lighter
3. Johnson was given a lighter

1. Johnson gave Trump a lighter.

$$\exists x((\text{give'}(e) \wedge \textsc{recipient}(e, \text{trump'}) \wedge \textsc{agent}(e, \text{johnson'}) \wedge \textsc{theme}(e, x) \wedge \text{lighter'}(x)))$$

$\rightarrow$ TRUTH VALUE is 0

Truth of these statements in our world model?

Remember the world where Trump gave Johnson a golden lighter? Are the following statements true in that world?

1. Johnson gave Trump a lighter
2. Trump gave Johnson a silver lighter
3. Johnson was given a lighter

1. Johnson gave Trump a lighter.

$$\exists x((\text{give'}(e) \wedge \textsc{recipient}(e, \text{trump'}) \wedge \textsc{agent}(e, \text{johnson'}) \wedge \textsc{theme}(e, x) \wedge \text{lighter'}(x)))$$

$\rightarrow$ TRUTH VALUE is 0

2. Trump gave Johnson a silver lighter

$$\exists x((\text{give'}(e) \wedge \textsc{agent}(e, \text{trump'}) \wedge \textsc{recipient}(e, \text{johnson'}) \wedge \textsc{theme}(e, x) \wedge \text{lighter'}(x) \wedge \text{silver'}(x)))$$

$\rightarrow$ TRUTH VALUE is 0

Truth of these statements in our world model?

Remember the world where Trump gave Johnson a golden lighter? Are the following statements true in that world?

1. Johnson gave Trump a lighter
2. Trump gave Johnson a silver lighter
3. Johnson was given a lighter

1. Johnson gave Trump a lighter.

$$\exists x((\text{give'}(e) \wedge \textsc{recipient}(e, \text{trump'}) \wedge \textsc{agent}(e, \text{johnson'}) \wedge \textsc{theme}(e, x) \wedge \text{lighter'}(x)))$$

$\rightarrow$ TRUTH VALUE is 0

2. Trump gave Johnson a silver lighter

$$\exists x((\text{give'}(e) \wedge \textsc{agent}(e, \text{trump'}) \wedge \textsc{recipient}(e, \text{johnson'}) \wedge \textsc{theme}(e, x) \wedge \text{lighter'}(x) \wedge \text{silver'}(x)))$$

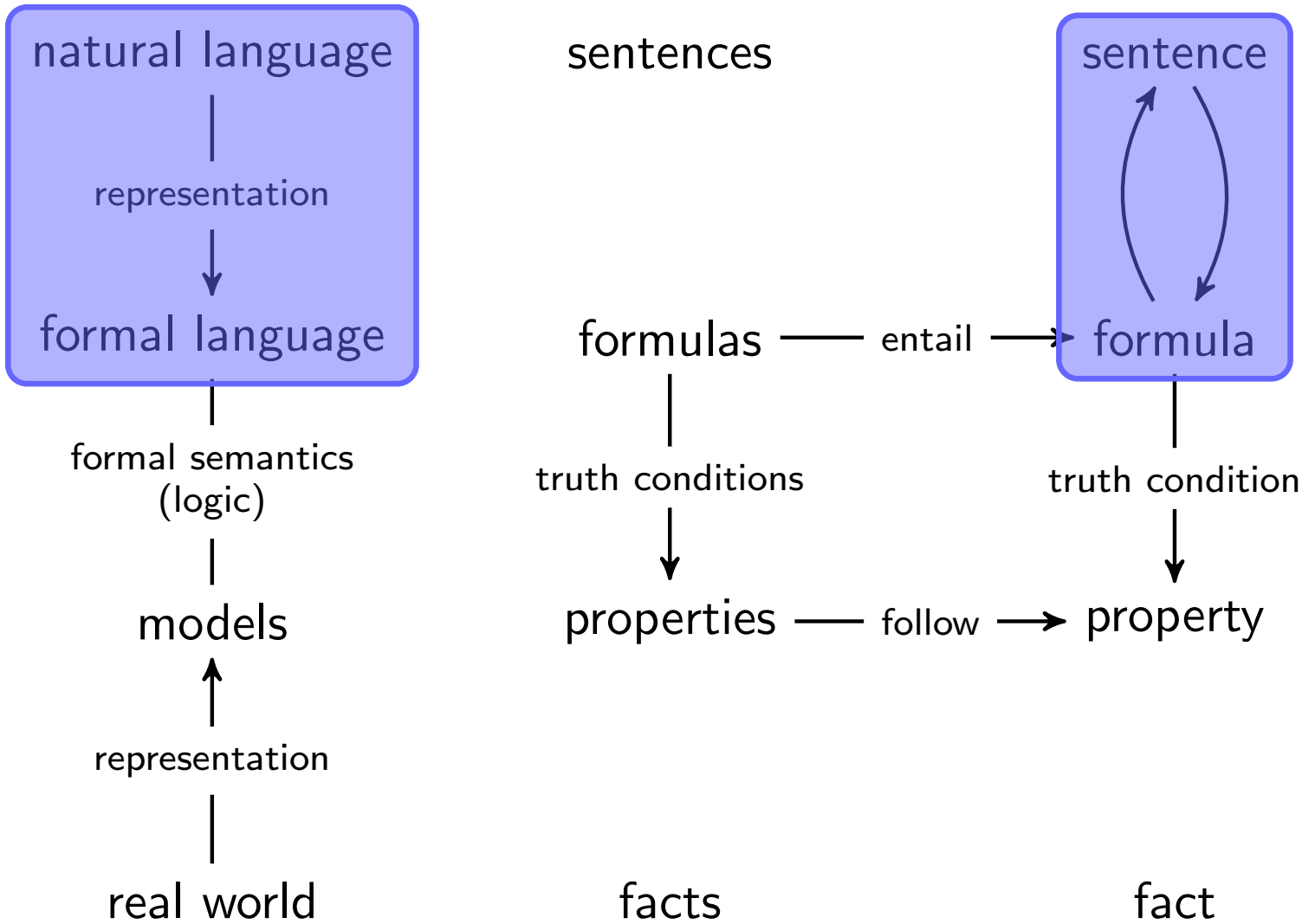
$\rightarrow$ TRUTH VALUE is 0

3. Johnson was given a lighter

$$\exists x((\text{give'}(e) \wedge \textsc{recipient}(e, \text{johnson'}) \wedge \textsc{theme}(e, x) \wedge \text{lighter'}(x)))$$

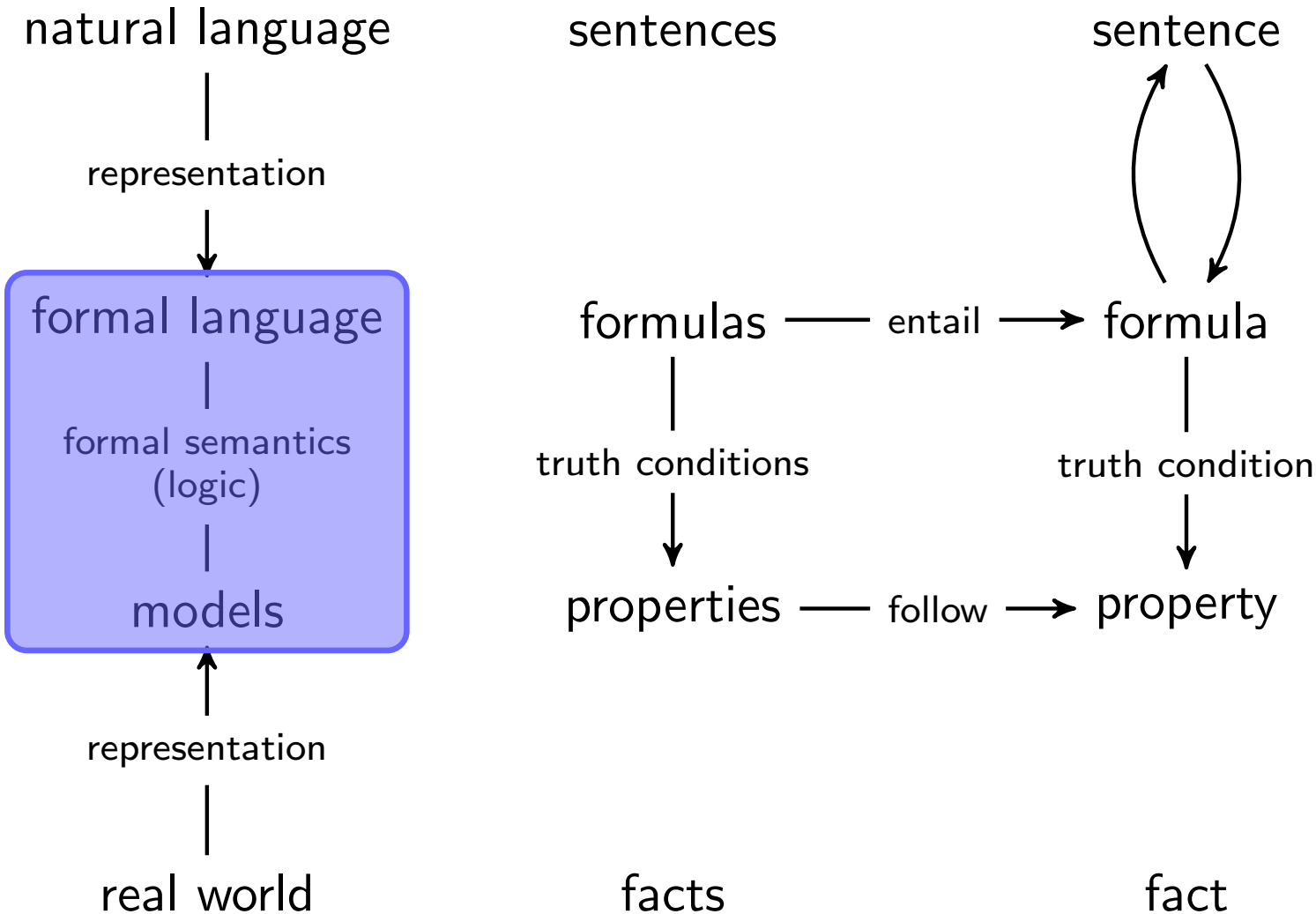
$\rightarrow$ TRUTH VALUE is 1

Pros: Precise; support composition; support reasoning



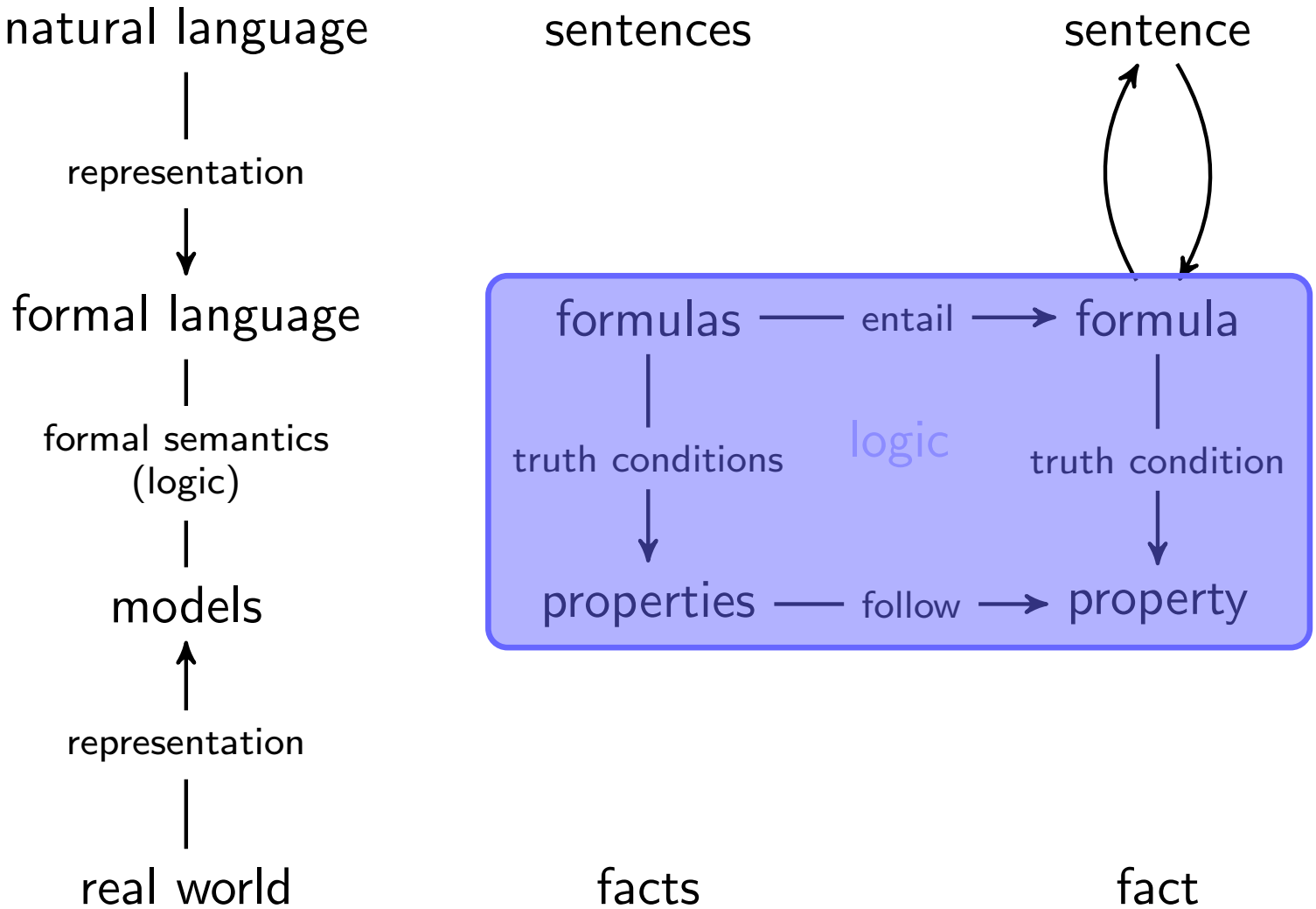
from Yanjing Wang

Pros: Precise; support composition; support reasoning



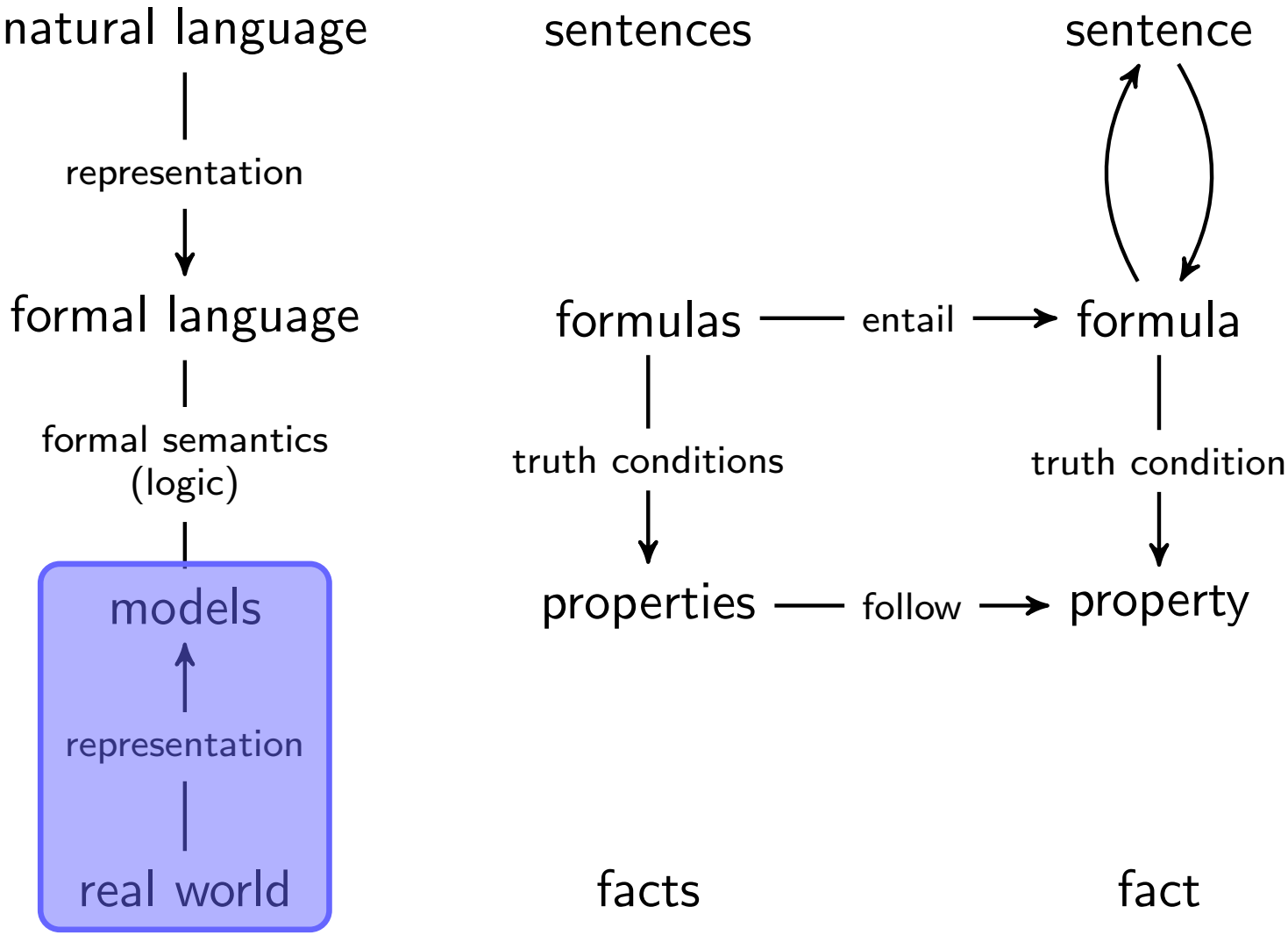
from Yanjing Wang

Pros: Precise; support composition; support reasoning



from Yanjing Wang

Pros: Precise; support composition; support reasoning



from Yanjing Wang

Reading

- Heim and Kratzer. *Semantics in Generative Grammar*. Chapter 1–3.