Case Study: Semantic Parsing as Code Generation

Srini Iyer, University of Washington

Semantic Parsing as Code Generation

Semantic Parsing as Code Generation

NL Query: `Adds a scalar to this vector in place.`

Semantic Parsing as Code Generation

NL Query: `Adds a scalar to this vector in place.`

```
public void add (final double arg0) {
  for (int loc0 = 0; loc0 < vecElements.length; loc0 ++) {
    vecElements[loc0] += arg0;
  }
}
```

Output Code

CONCODE DATASET (Iyer et al., 2018)

Why generate Code?

- Semantic Parsing for Programmers
- Inexpensive datasets not requiring annotation
- Full expressivity not limited by logical forms/DSL

Why generate Code?

- Semantic Parsing for Programmers
- Inexpensive datasets not requiring annotation
- Full expressivity not limited by logical forms/DSL

- Smarter IDEs based on Natural Language!
- Extending Projects using NL Specifications
- Natural Language based Code Retrieval

Why generate Code?

- Semantic Parsing for Programmers
- Inexpensive datasets not requiring annotation
- Full expressivity not limited by logical forms/DSL

GitHub

Why generate Code?

- Semantic Parsing for Programmers
- Inexpensive datasets not requiring annotation
- Full expressivity not limited by logical forms/DSL

GitHub

- Code Comments
- Code Documentation
- Commit Messages

Why generate Code?

- Semantic Parsing for Programmers
- Inexpensive datasets not requiring annotation
- Full expressivity not limited by logical forms/DSL

GitHub

- Code Comments
- Code Documentation
- Commit Messages

stackoverflow

stackoverflow

Compute the actual textwidth inside a textblock

Solution working on WP8 device

5

```c#
public int TextWidth(string text) {
  TextBlock t = new TextBlock();
  t.Text = text;
  return (int)Math.Ceiling(t.ActualWidth);
}
```

c#

According to all information I found, ActualWidth should not be set until control is measured …….

Why generate Code?

- Semantic Parsing for Programmers
- Inexpensive datasets not requiring annotation
- Full expressivity not limited by logical forms/DSL

GitHub

- Hearthstone (Ling et al. 2015)
- MTG (Ling et al. 2015)
- CONCODE (Iyer et al. 2018)
- PYTHON (Barone et al., 2017)

stackoverflow

- CODENN (Iyer et al. 2016)
- StaQC (Yao et al. 2018)
- CONALA (Yin et al. 2018)

Why generate Code?

- Semantic Parsing for Programmers
- Inexpensive datasets not requiring annotation
- Full expressivity not limited by logical forms/DSL

```
λx.river(x)
∧ ∃y.state(y)
∧ next_to(y,'Texas')
∧ loc(x, y)
```

Which rivers run through the states bordering Texas?

```
SELECT river_name FROM river
WHERE traverse IN (SELECT border_info.border
   FROM border_info WHERE
```

Directly generate Programming Source Code

- Single Line of Source Code 
- Entire Method 
- Entire Class 

Directly generate Programming Source Code

- Single Line of Source Code
- Entire Method
- Entire Class

Directly generate Programming Source Code

- Single Line of Source Code
- Entire Method
- Entire Class

DJANGO DATASET (Oda et al., 2015.)

Delete the value under the key key of the self._expire_info dictionary.

```
del self . _expire_info [ key ]
```

Directly generate Programming Source Code

- Single Line of Source Code
- Entire Method
- Entire Class

DJANGO DATASET (Oda et al., 2015.)

Delete the value under the key key of the self._expire_info dictionary.

```
del self . _expire_info [ key ]
```

NL2BASH DATASET (Lin et al., 2018)

Search for the string 'git' in all the files under current directory tree without traversing into '.git' folder and excluding files that have 'git' in their names

```
find . -not -name ".git" -not -path "*.git*" -not -name
"*git*" | xargs -I {} grep git {}
```

Directly generate Programming Source Code

- Single Line of Source Code
- Entire Method
- Entire Class

SCHOLAR DATASET (Iyer et al., 2017)

How many people have collaborated with Mark Steedman ?

```
SELECT count(DISTINCT author.authorId)
FROM writes, author
WHERE writes.authorId = author.authorId
AND author.authorName != “Mark Steedman"
AND writes.paperId IN
  (SELECT writes.paperId
   FROM writes, author
   WHERE writes.authorId = author.authorId
   AND author.authorName = “Mark Steedman”);
```

Directly generate Programming Source Code

- Single Line of Source Code
- Entire Method
- Entire Class

CONCODE DATASET (Iyer et al., 2018)

```
double[] vecElements;
double[] weights;
public void inc() ;
```

Environment

```
Adds a scalar to this vector in place.
```

NL

Directly generate Programming Source Code

- Single Line of Source Code
- Entire Method
- Entire Class

CONCODE DATASET (Iyer et al., 2018)

```
double[] vecElements;
double[] weights;
public void inc() ;
```

Environment

```
Adds a scalar to this vector in place.
```

NL

Input

```
public void add (final double arg0) {
  for (int loc0 = 0; loc0 < vecElements.length; loc0 ++) {
    vecElements[loc0] += arg0;
  }
}
```

Output Code

Directly generate Programming Source Code

- Single Line of Source Code
- Entire Method
- Entire Class

CONCODE DATASET (Iyer et al., 2018)

```
double[] vecElements;
double[] weights;
public void inc() ;
```

Environment

```
Adds a scalar to this vector in place.
```

NL

Input

```
public void add (final double arg0) {
  for (int loc0 = 0; loc0 < vecElements.length; loc0 ++) {
    vecElements[loc0] += arg0;
  }
}
```

Output Code

Directly generate Programming Source Code

- Single Line of Source Code
- Entire Method
- Entire Class

CONCODE DATASET (Iyer et al., 2018)

```
double[] vecElements;
double[] weights;
public void inc() ;
```

Environment

```
Adds a scalar to this vector in place.
```

NL

Input

```
public void add (final double arg0) {
  for (int loc0 = 0; loc0 < vecElements.length; loc0 ++) {
    vecElements[loc0] += arg0;
  }
}
```

Output Code

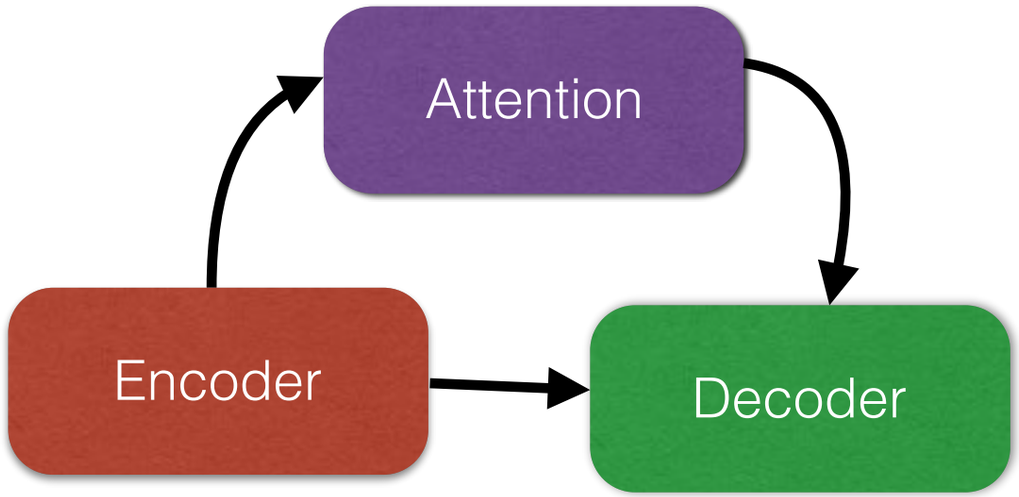
Directly generate Programming Source Code

- Single Line of Source Code
- Entire Method
- Entire Class

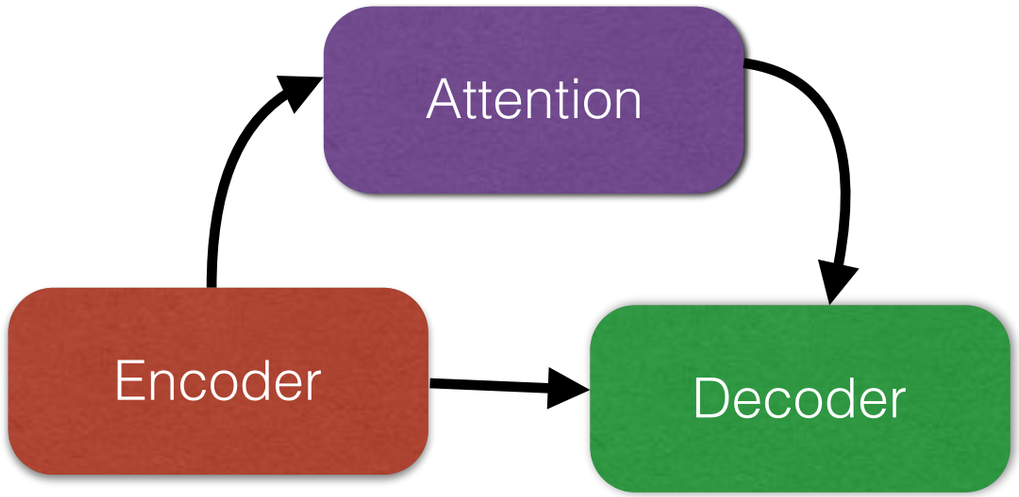
HearthStone DATASET (Ling et al., 2016)


```
class DivineFavor(SpellCard):
  def __init__(self):
    super().__init__(
      "Divine Favor",
      3,
      CHARACTER_CLASS.PALADIN,
      CARD_RARITY.RARE)

def use(self, player, game):
  super().use(player, game)
  difference =
    len(game.other_player.hand) - len(player.hand)
  for i in range(0, difference):
    player.draw()
```

Encoder-Decoder Models

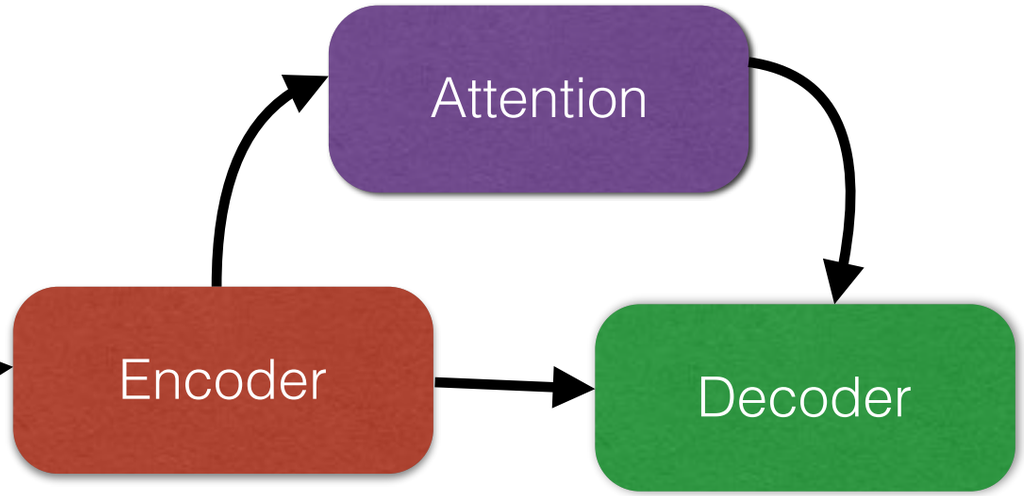


Encoder-Decoder Models



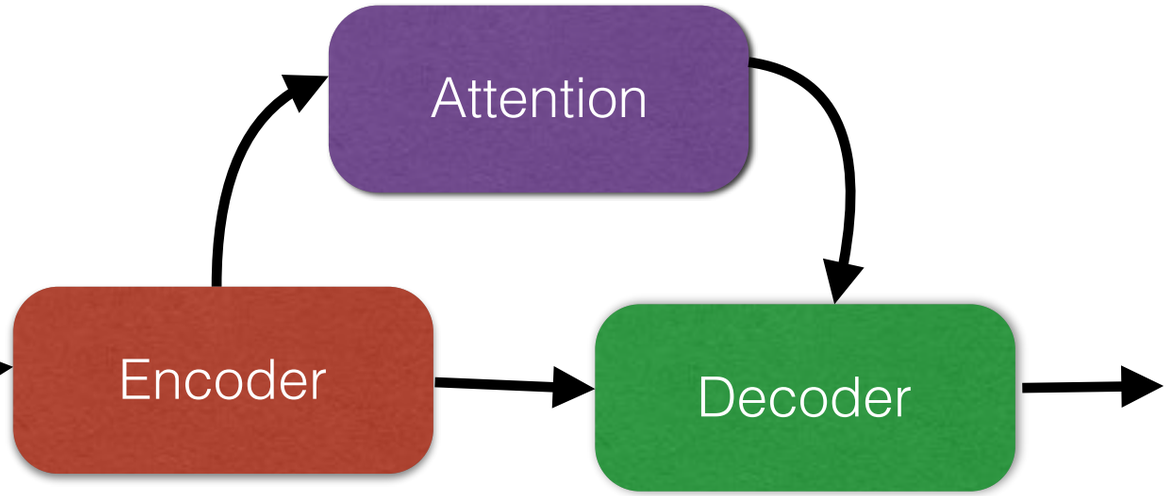

```
double[] vecElements;
double[] weights;
public void inc() ;

Adds a scalar to this
vector in place.
```

Encoder-Decoder Models



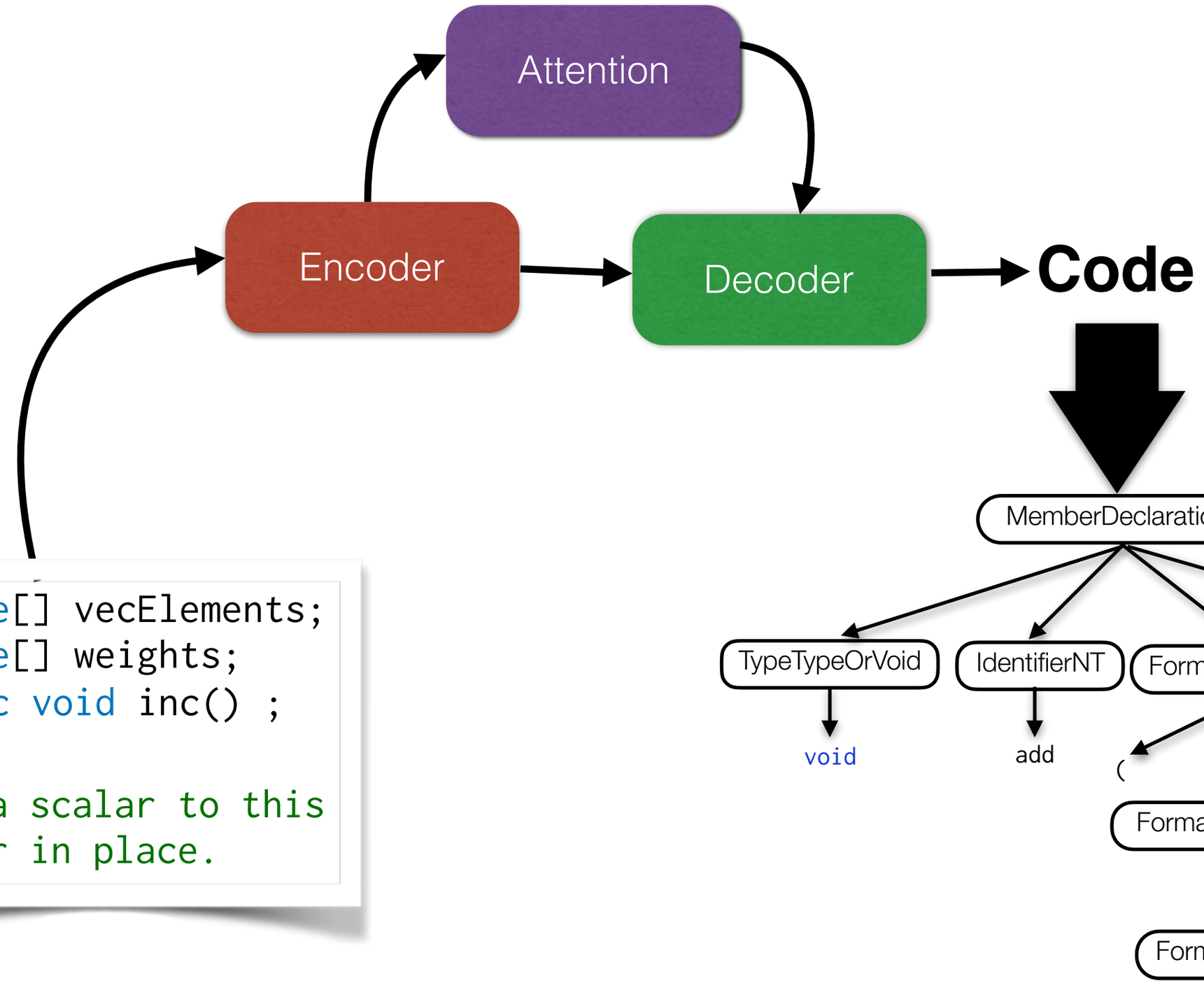

```
double[] vecElements;
double[] weights;
public void inc() ;

Adds a scalar to this
vector in place.
```

Encoder-Decoder Models




```
double[] vecElements;
double[] weights;
public void inc() ;

Adds a scalar to this
vector in place.
```

Encoder-Decoder Models




```
double[] vecElements;
double[] weights;
public void inc() ;

Adds a scalar to this
vector in place.
```

Input Components

Input Components

Text Components

Adjacent Minions have +1 attack

(Hearthstone, Ling et al., 2016)

Input Components

Text Components

Adjacent Minions have +1 attack

(Hearthstone, Ling et al., 2016)

Structured inputs:

Name: DireWolfAlpha

Health: 2

(Hearthstone, Ling et al., 2016)

Input Components

Text Components

Adjacent Minions have +1 attack

(Hearthstone, Ling et al., 2016)

Structured inputs:

Name: DireWolfAlpha

Health: 2

(Hearthstone, Ling et al., 2016)

Code Context:

```
double[] vecElements;
double[] weights;
public void inc() ;
```

(CONCODE dataset, Iyer et al., 2018)

Input Components

Text Components

Adjacent Minions have +1 attack

(Hearthstone, Ling et al., 2016)

Structured inputs:

Name: DireWolfAlpha

Health: 2

(Hearthstone, Ling et al., 2016)

Code Context:

```
double[] vecElements;
double[] weights;
public void inc() ;
```

(CONCODE dataset, Iyer et al., 2018)

```
SELECT DISTINCT flight.flight id
WHERE city.city name = 'SEATTLE' ...
date day.year = 1993 AND date day.month number = 2 ...
```

(ATIS dataset, Suhr et al., 2018)

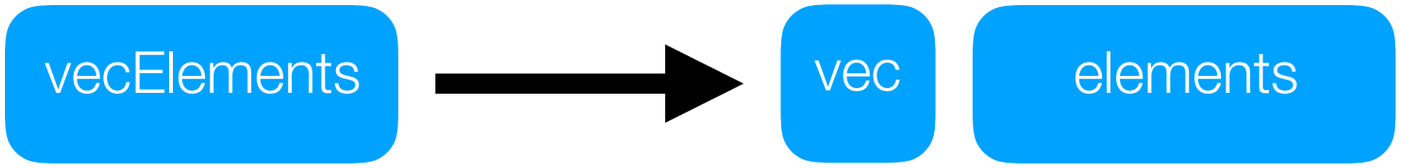
Preprocessing

Preprocessing

- Splitting entities by camel casing

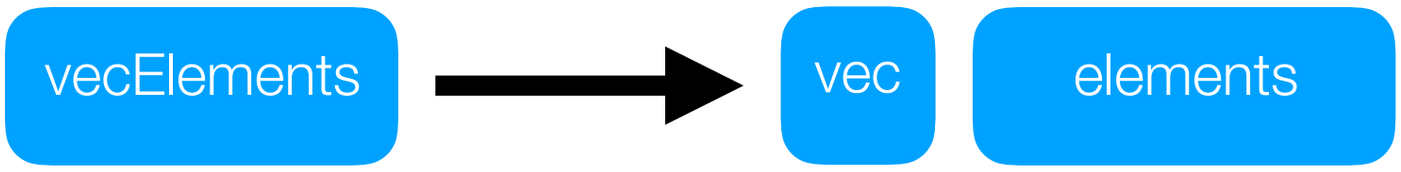
Preprocessing

- Splitting entities by camel casing **(Iyer et al., 2018)**



Preprocessing

- Splitting entities by camel casing **(Iyer et al., 2018)**



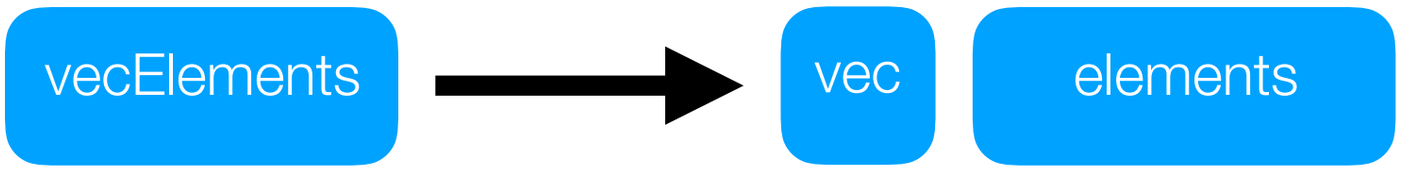
- Entity Anonymization **(Iyer at al., 2017, Suhr at al., 2018)**

```
show me flights from seattle to boston next monday
```

```
SELECT DISTINCT flight.flight id ...
city.city name = 'SEATTLE' ... city.city name = 'BOSTON' ...
date day.year = 1993 AND date day.month number = 2
AND date day.day number = 8 ...
```

Preprocessing

- Splitting entities by camel casing **(Iyer et al., 2018)**



- Entity Anonymization **(Iyer at al., 2017, Suhr at al., 2018)**

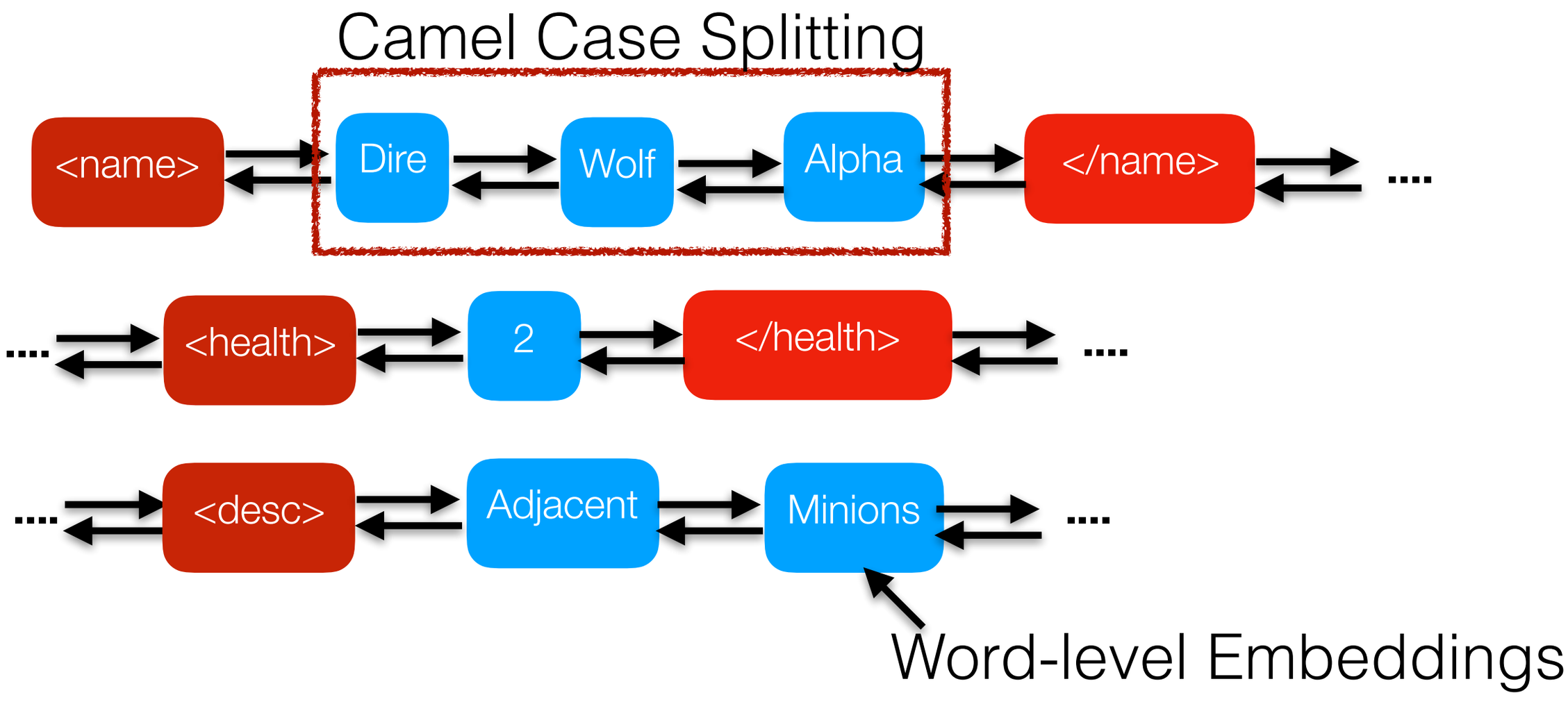
```
show me flights from seattle to boston next monday

SELECT DISTINCT flight.flight id ...
city.city name = 'SEATTLE' ... city.city name = 'BOSTON' ...
date day.year = 1993 AND date day.month number = 2
AND date day.day number = 8 ...

show me flights from CITY#1 to CITY#2 DAY#1 MONTH#1 YEAR#1

SELECT DISTINCT flight.flight id ...
city.city name = CITY#1 ... city.city name
= CITY#2 ... date day.year = YEAR#1
AND date day.month number = MONTH#1
AND date day.day number = DAY#1 ...
```

Encoding

- Sequential BiLSTM (Yin et al., 2017)
- Component-wise BiLSTM (Rabinovich et al., 2017, Iyer et al., 2018)
- Character-to-word Encoder (Ling et al., 2016)

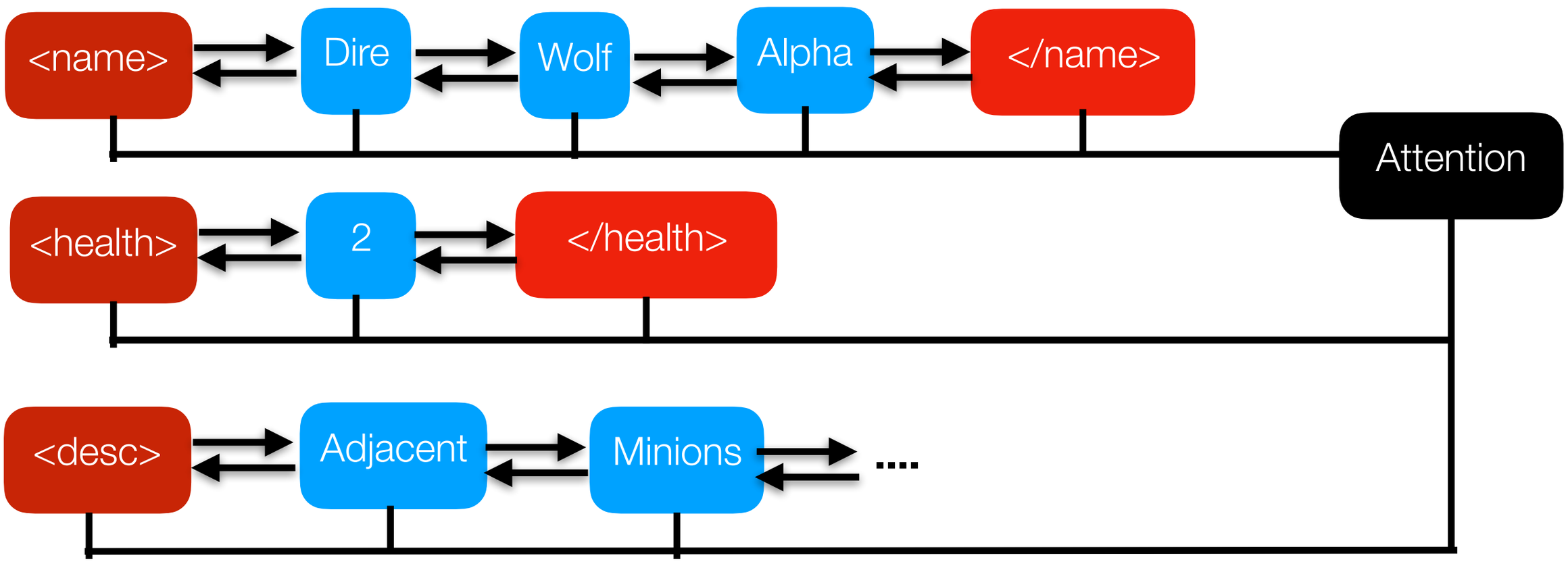
Encoding

- Sequential BiLSTM (Yin et al., 2017)
- Component-wise BiLSTM (Rabinovich et al., 2017, Iyer et al., 2018)
- Character-to-word Encoder (Ling et al., 2016)



Encoding

- Sequential BiLSTM (Yin et al., 2017)
- Component-wise BiLSTM (Rabinovich et al., 2017, Iyer et al., 2018)
- Character-to-word Encoder (Ling et al., 2016)

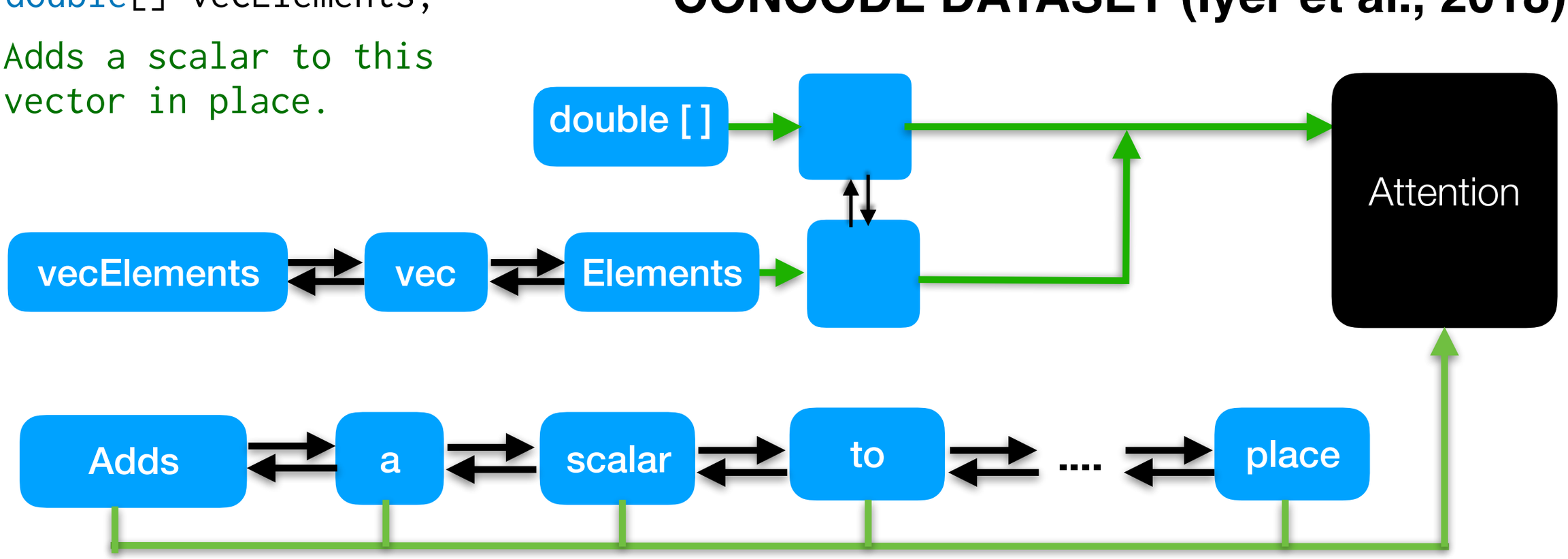


Encoding

- Sequential BiLSTM (Yin et al., 2017)
- Component-wise BiLSTM (Rabinovich et al., 2017, Iyer et al., 2018)
- Character-to-word Encoder (Ling et al., 2016)

```
double[] vecElements;
Adds a scalar to this
vector in place.
```

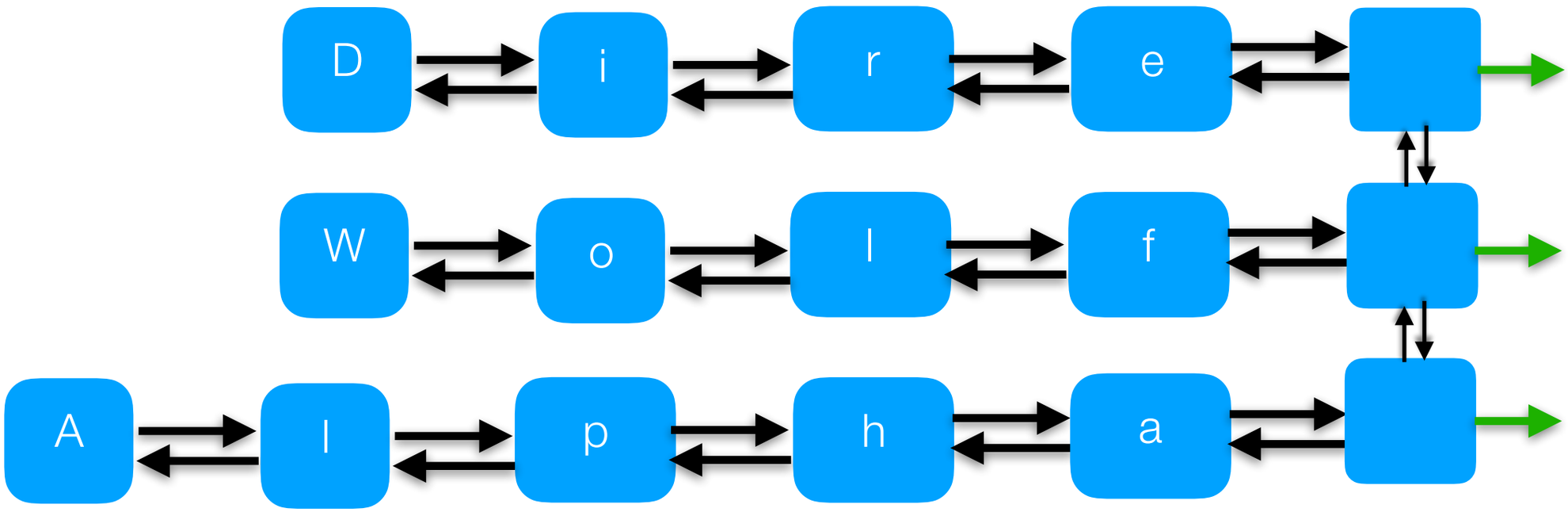
CONCODE DATASET (Iyer et al., 2018)



NL Query

Encoding

- Sequential BiLSTM (Yin et al., 2017)
- Component-wise BiLSTM (Rabinovich et al., 2017, Iyer et al., 2018)
- Character-to-word Encoder (Ling et al., 2016)



Compositional Word Embeddings using Characters

Decoding

Decoding

- Token-Sequence based Decoder (Ling et al., 2016)

Decoding

- Token-Sequence based Decoder (Ling et al., 2016)
  - Can generate syntactically ill-formed code

Decoding

- Token-Sequence based Decoder (Ling et al., 2016)
  - Can generate syntactically ill-formed code
- Production Rules based Decoder (Yin et al., 2017, Iyer et al., 2018)

Decoding

- Token-Sequence based Decoder (Ling et al., 2016)
  - Can generate syntactically ill-formed code
- Production Rules based Decoder (Yin et al., 2017, Iyer et al., 2018)
- Abstract Syntax Networks (Rabinovich et al., 2017)

Decoding

- Token-Sequence based Decoder (Ling et al., 2016)
  - Can generate syntactically ill-formed code
- Production Rules based Decoder (Yin et al., 2017, Iyer et al., 2018)
- Abstract Syntax Networks (Rabinovich et al., 2017)

Guarantee Syntax Correctness

Decoding

- Token-Sequence based Decoder (Ling et al., 2016)
  - Can generate syntactically ill-formed code
- Production Rules based Decoder (Yin et al., 2017, Iyer et al., 2018)
- Abstract Syntax Networks (Rabinovich et al., 2017)

Generating Grammar Rules

```
public void add(final double arg0) {
  .......
}
```

Yin et al., 2017. Iyer et al., 2018.

Generating Grammar Rules

```
public void add(final double arg0) {
  .......
}
```

```
MethodDeclaration -->
  TypeTypeOrVoid IdentifierNT FormalParameters MethodBody
TypeTypeOrVoid --> void
IdentifierNT --> add
FormalParameters --> ( FormalParameterList )
FormalParameterList --> FormalParameter
FormalParameter --> Star_21 TypeType VariableDeclaratorId
Star_21 --> VariableModifier
VariableModifier --> final
TypeType --> Nt_41
Nt_41 --> PrimitiveType
PrimitiveType --> double
VariableDeclaratorId --> IdentifierNT
IdentifierNT --> arg0
MethodBody --> Block
Block --> { }
```

Yin et al., 2017. Iyer et al., 2018.

Generating Grammar Rules

```
public void add(final double arg0) {
  .......
}
```

```
MethodDeclaration -->
  TypeTypeOrVoid IdentifierNT FormalParameters MethodBody
TypeTypeOrVoid --> void
IdentifierNT --> add
FormalParameters --> ( FormalParameterList )
FormalParameterList --> FormalParameter
FormalParameter --> Star_21 TypeType VariableDeclaratorId
Star_21 --> VariableModifier
VariableModifier --> final
TypeType --> Nt_41
Nt_41 --> PrimitiveType
PrimitiveType --> double
VariableDeclaratorId --> IdentifierNT
IdentifierNT --> arg0
MethodBody --> Block
Block --> { }
```

Yin et al., 2017. Iyer et al., 2018.

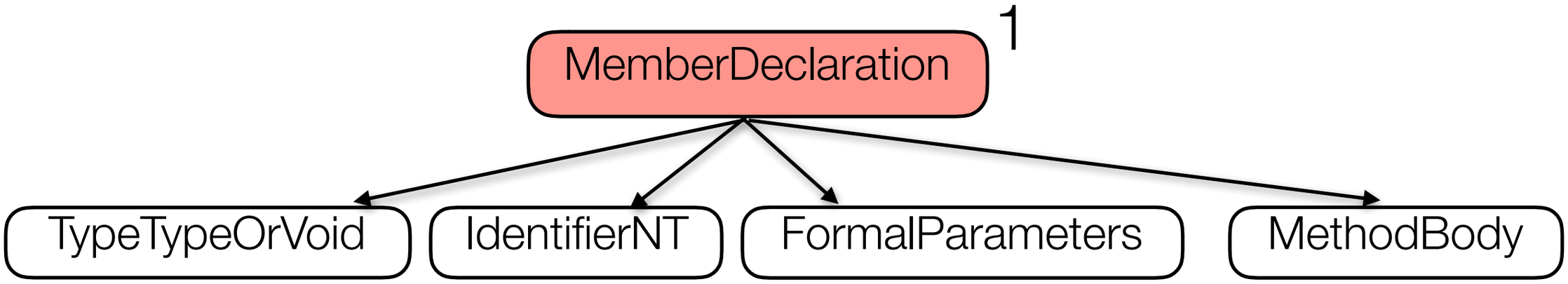
Generating Grammar Rules

MemberDeclaration [1]

```
public void add(final double arg0) {
  .......
}
```

Yin et al., 2017. Iyer et al., 2018.

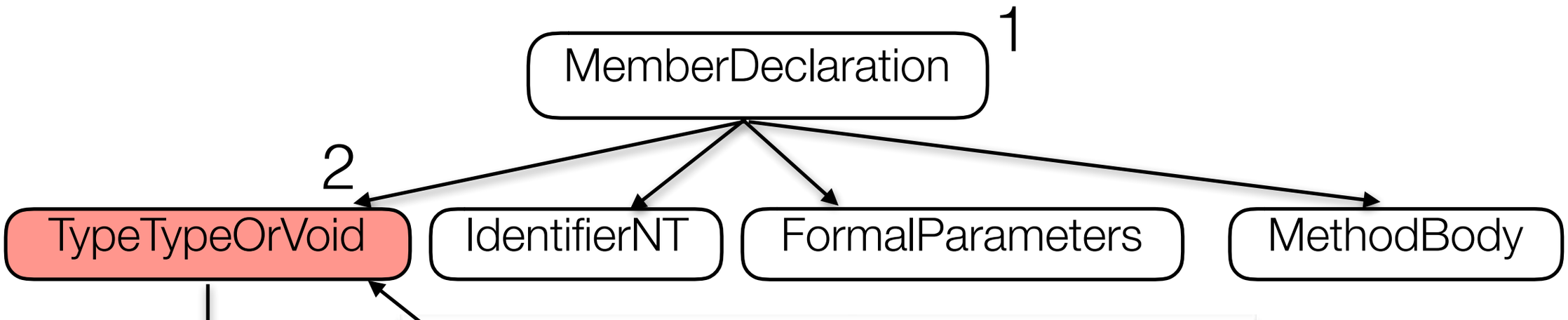
Generating Grammar Rules




```
public void add(final double arg0) {
  .......
}
```

Yin et al., 2017. Iyer et al., 2018.

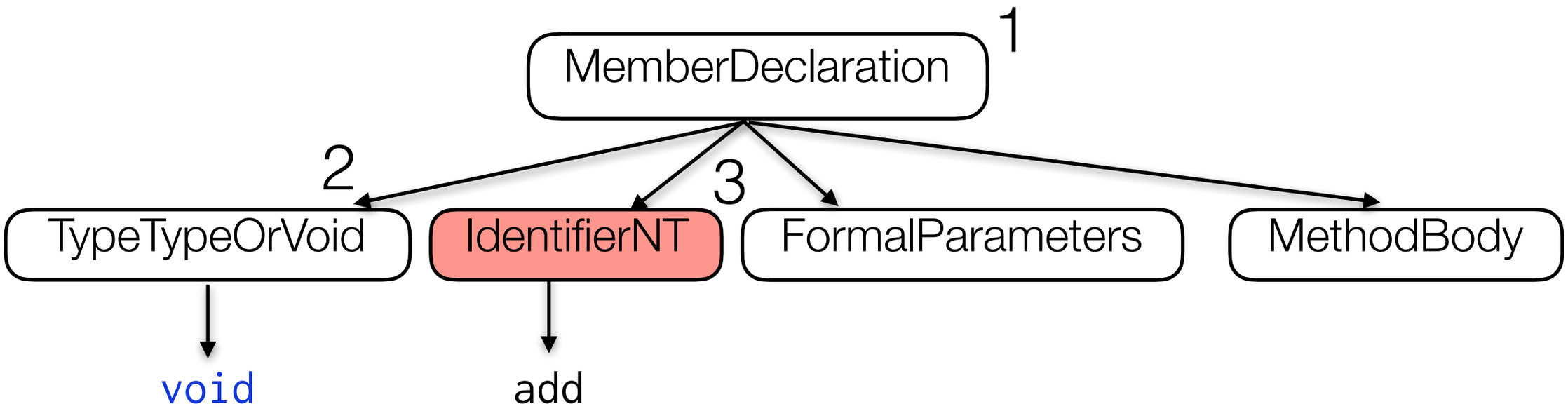
Generating Grammar Rules




```
public void add(final double arg0) {
  .......
}
```

Yin et al., 2017. Iyer et al., 2018.

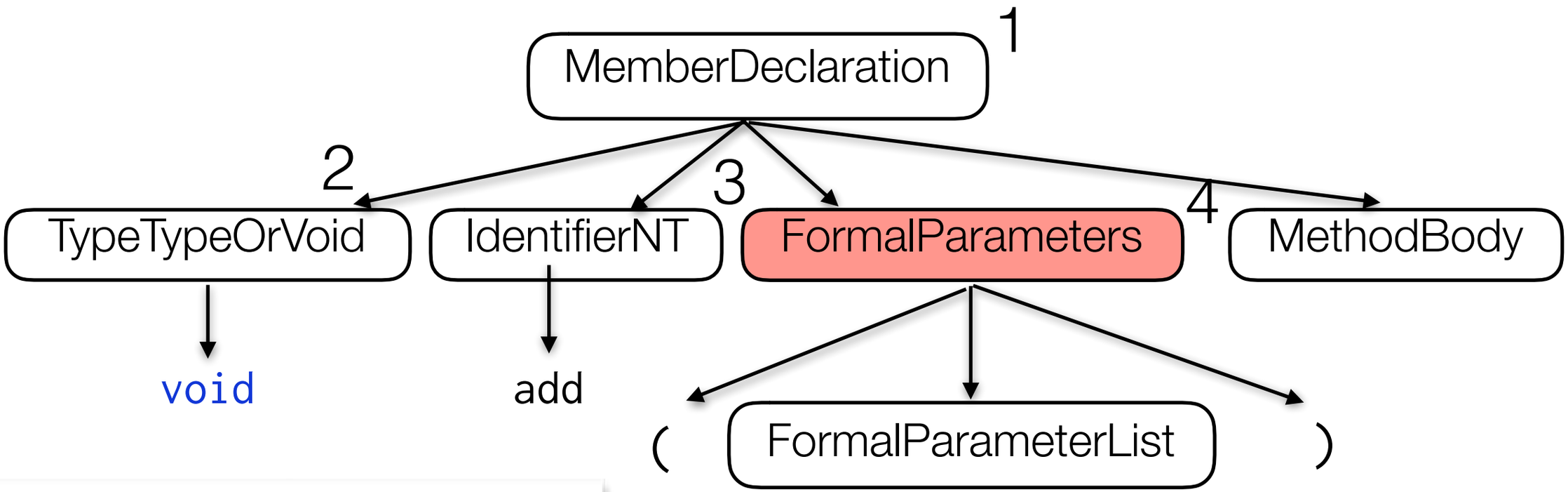
Generating Grammar Rules




```
public void add(final double arg0) {
  .......
}
```

Yin et al., 2017. Iyer et al., 2018.

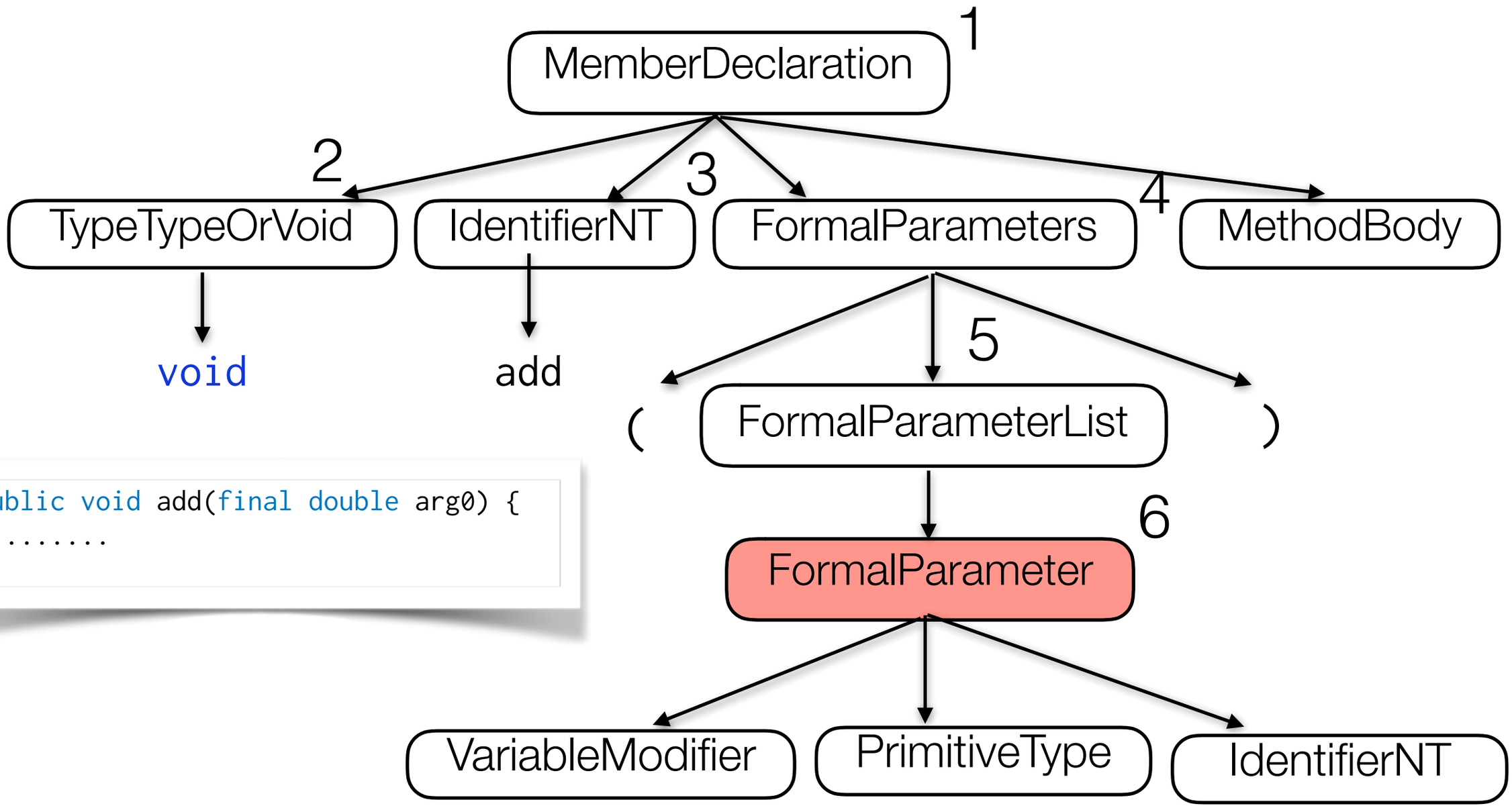
Generating Grammar Rules




```
public void add(final double arg0) {
  .......
}
```

Yin et al., 2017. Iyer et al., 2018.

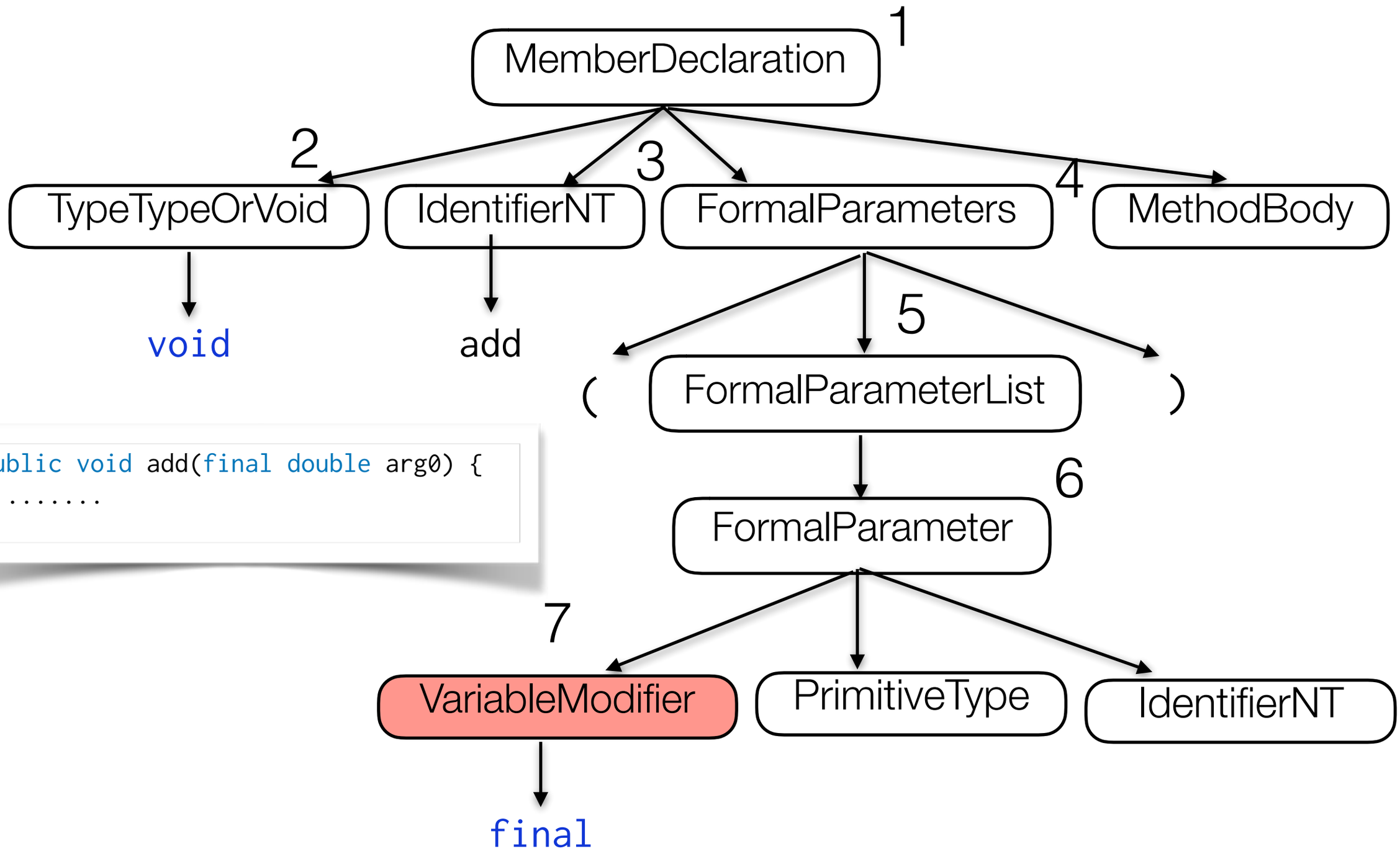
Generating Grammar Rules




```
public void add(final double arg0) {
  .......
}
```

Yin et al., 2017. Iyer et al., 2018.

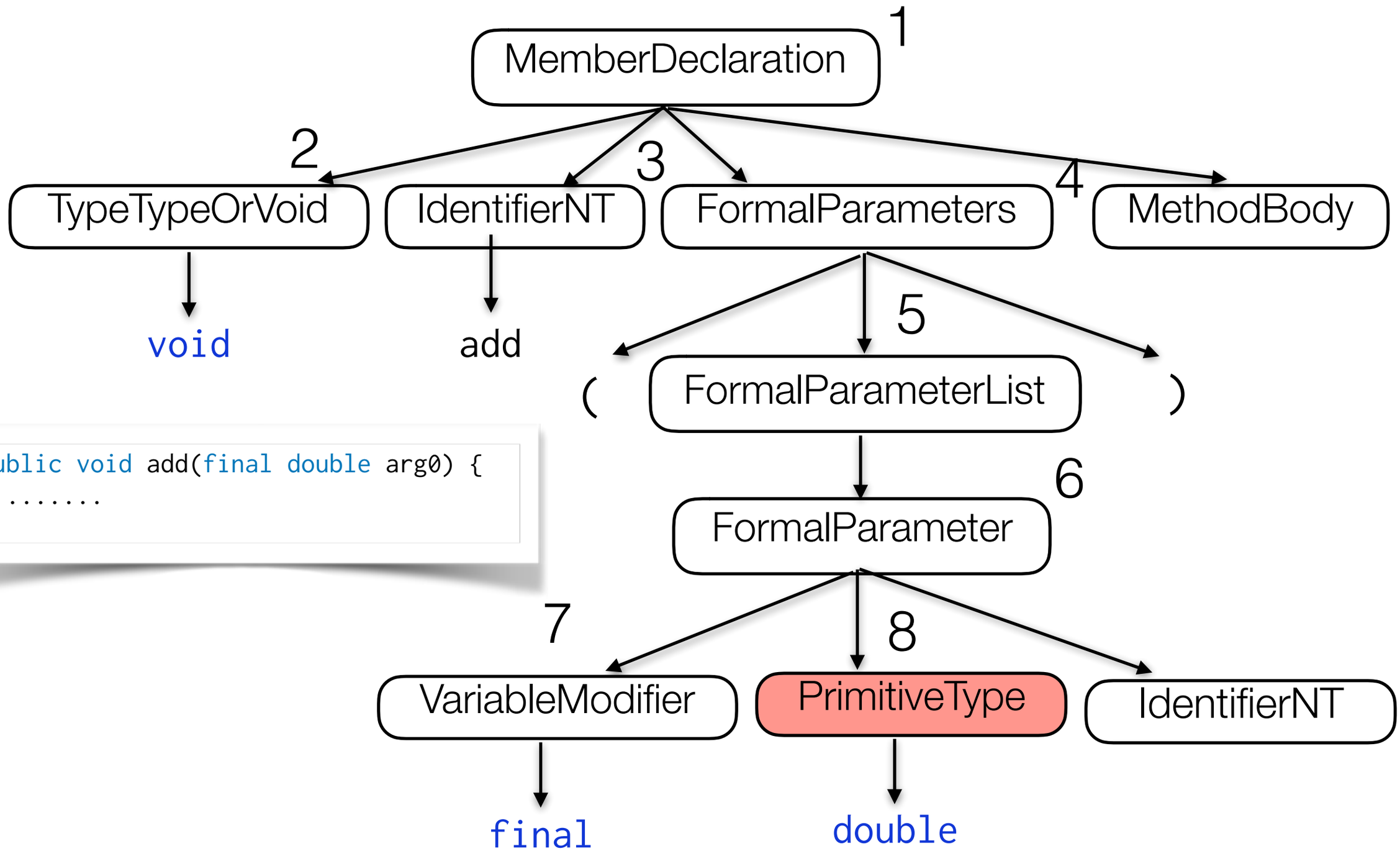
Generating Grammar Rules




```
public void add(final double arg0) {
  .......
}
```

Yin et al., 2017. Iyer et al., 2018.

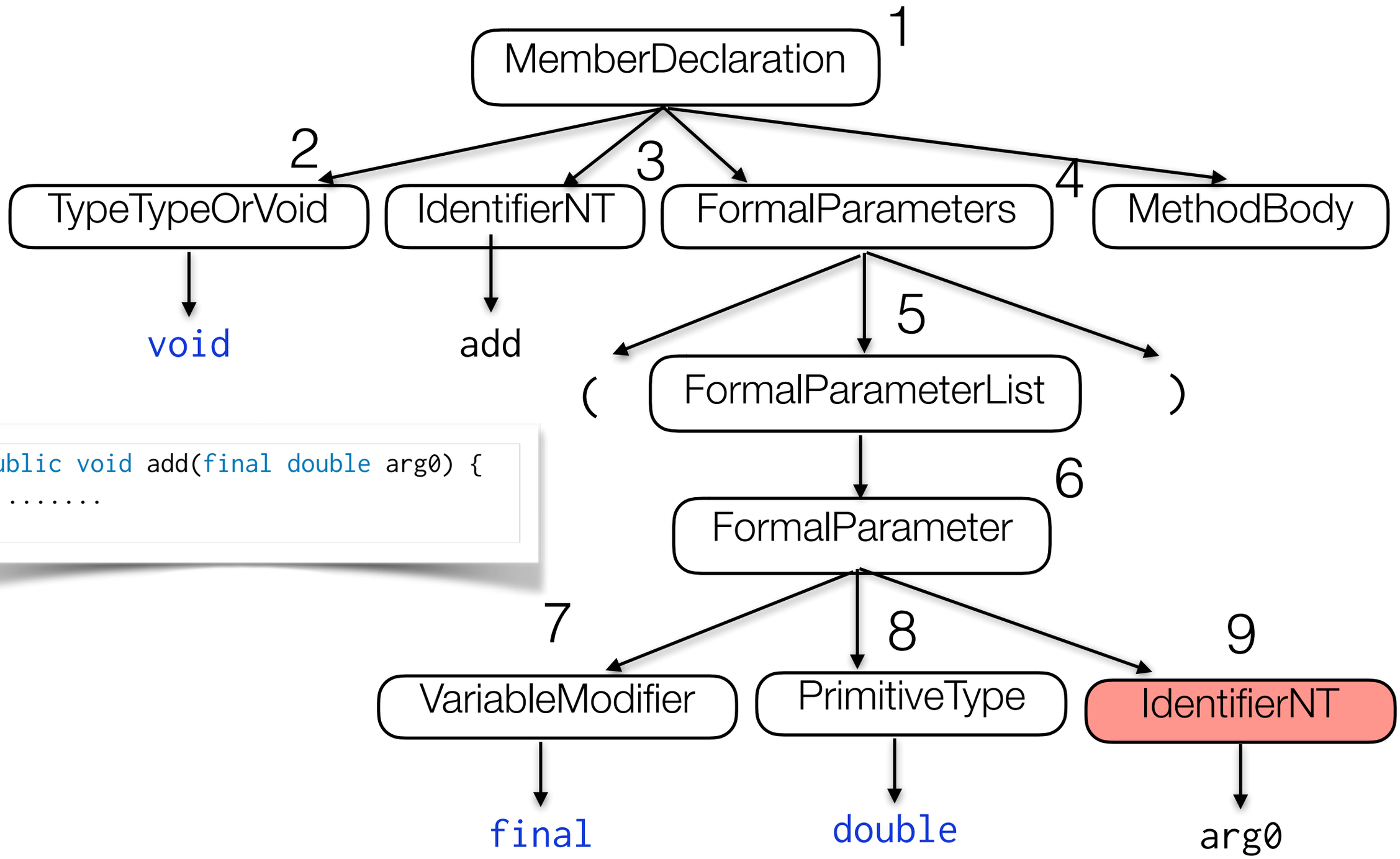
Generating Grammar Rules




```
public void add(final double arg0) {
  .......
}
```

Yin et al., 2017. Iyer et al., 2018.

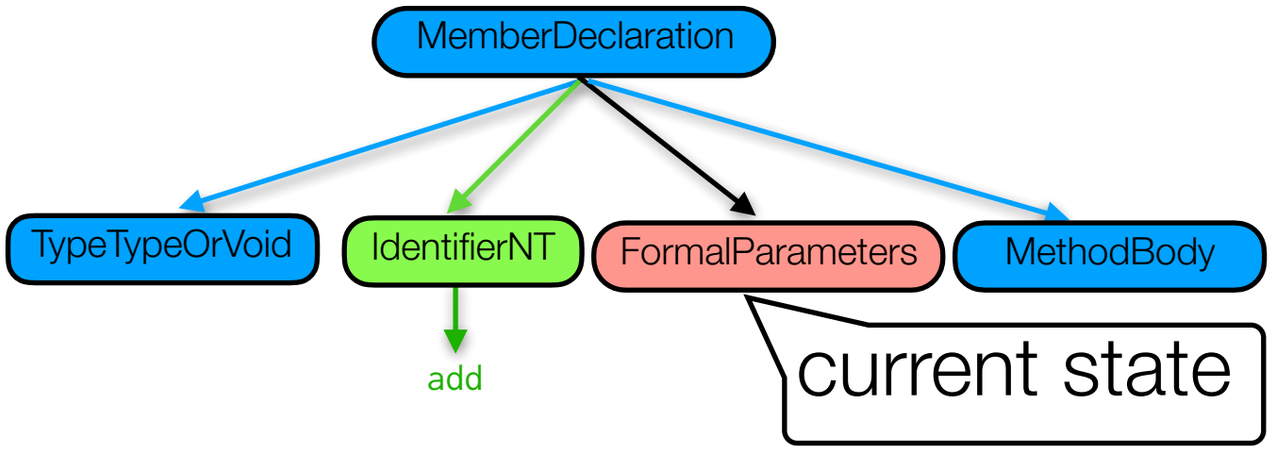
Generating Grammar Rules




```
public void add(final double arg0) {
  .......
}
```

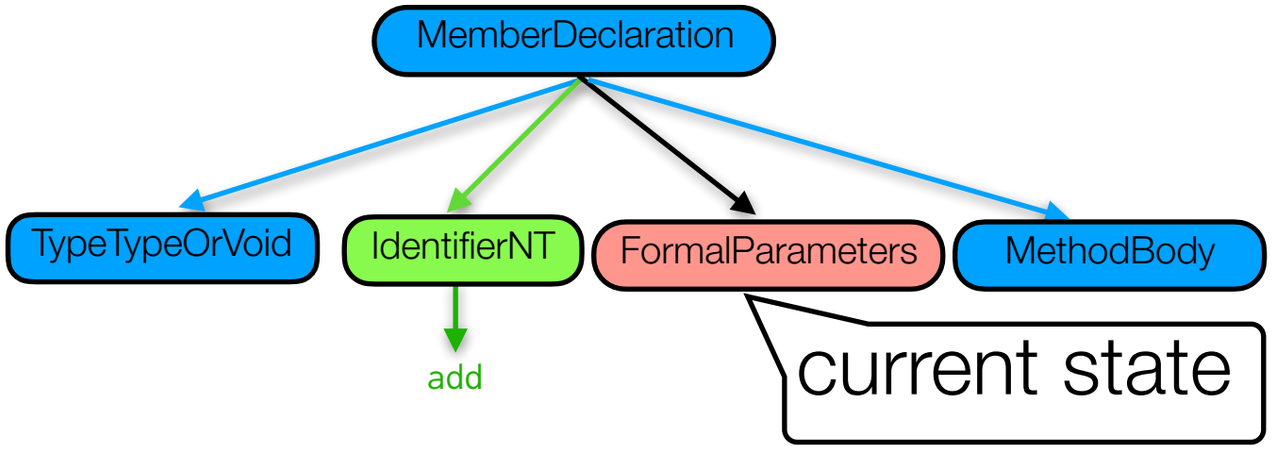
Yin et al., 2017. Iyer et al., 2018.

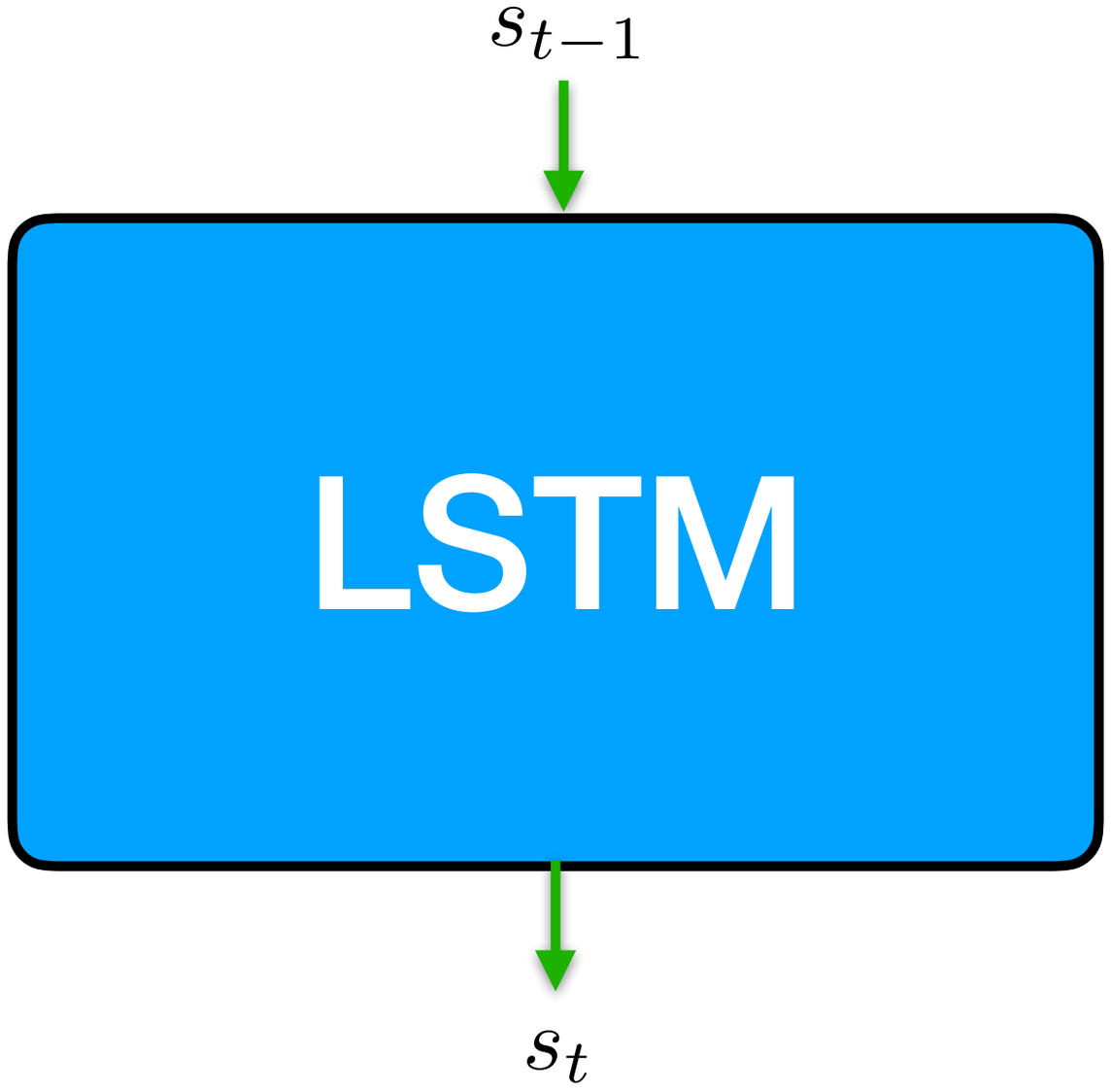
Decoder RNN



Iyer et al., 2018. Yin et al., 2017.

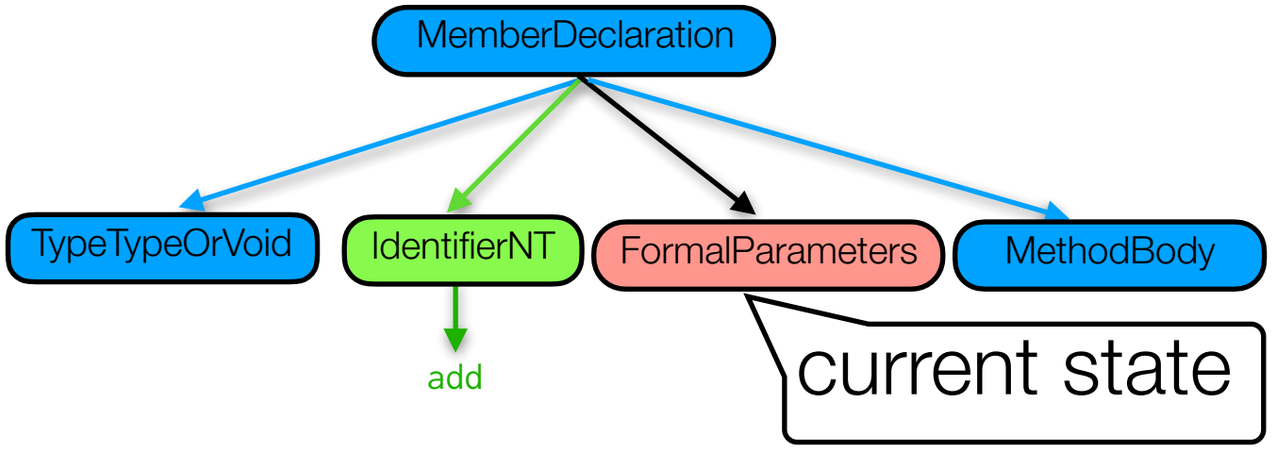
Decoder RNN



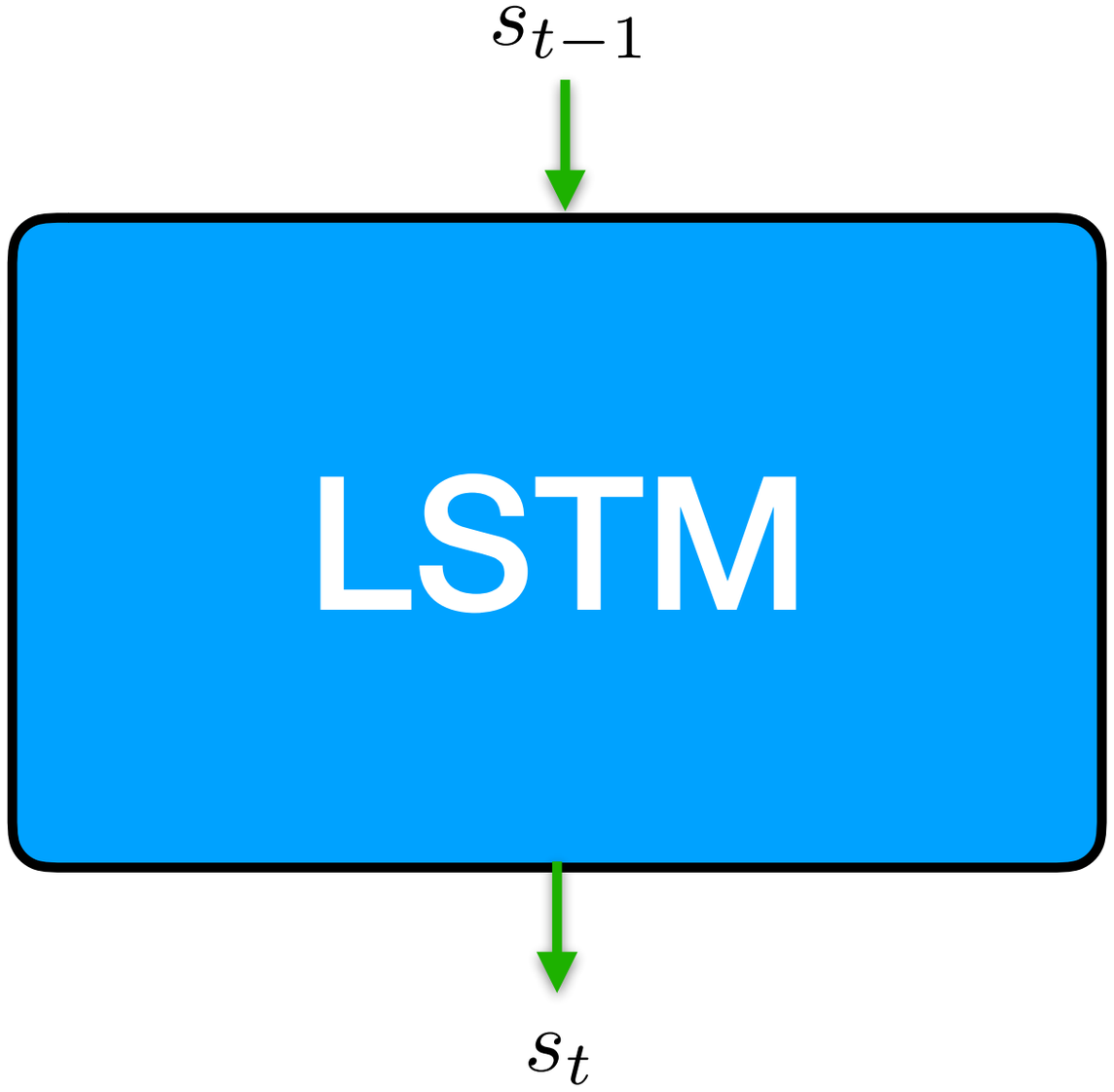


Iyer et al., 2018. Yin et al., 2017.

Decoder RNN

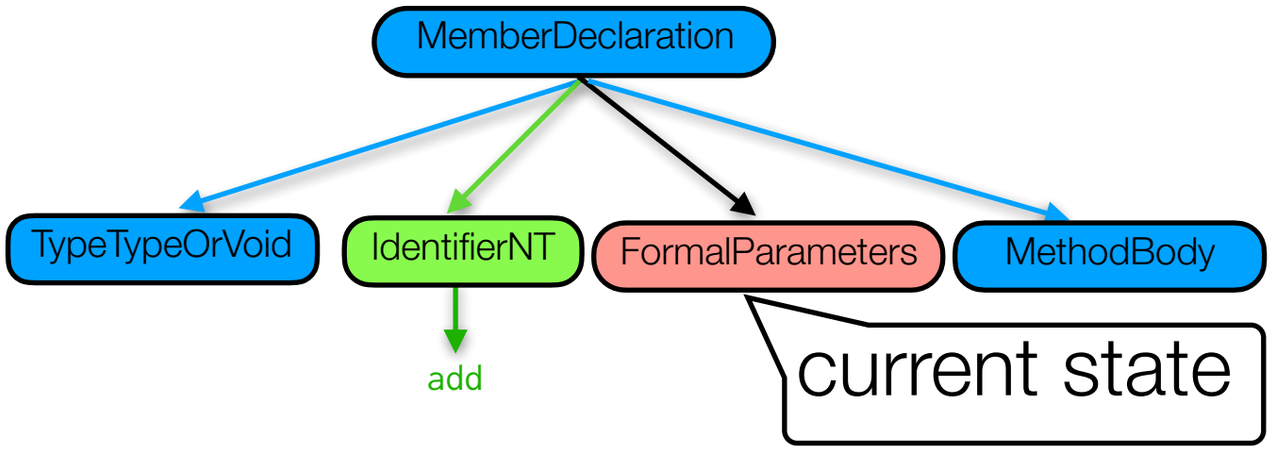


FormalParameters (n_t)



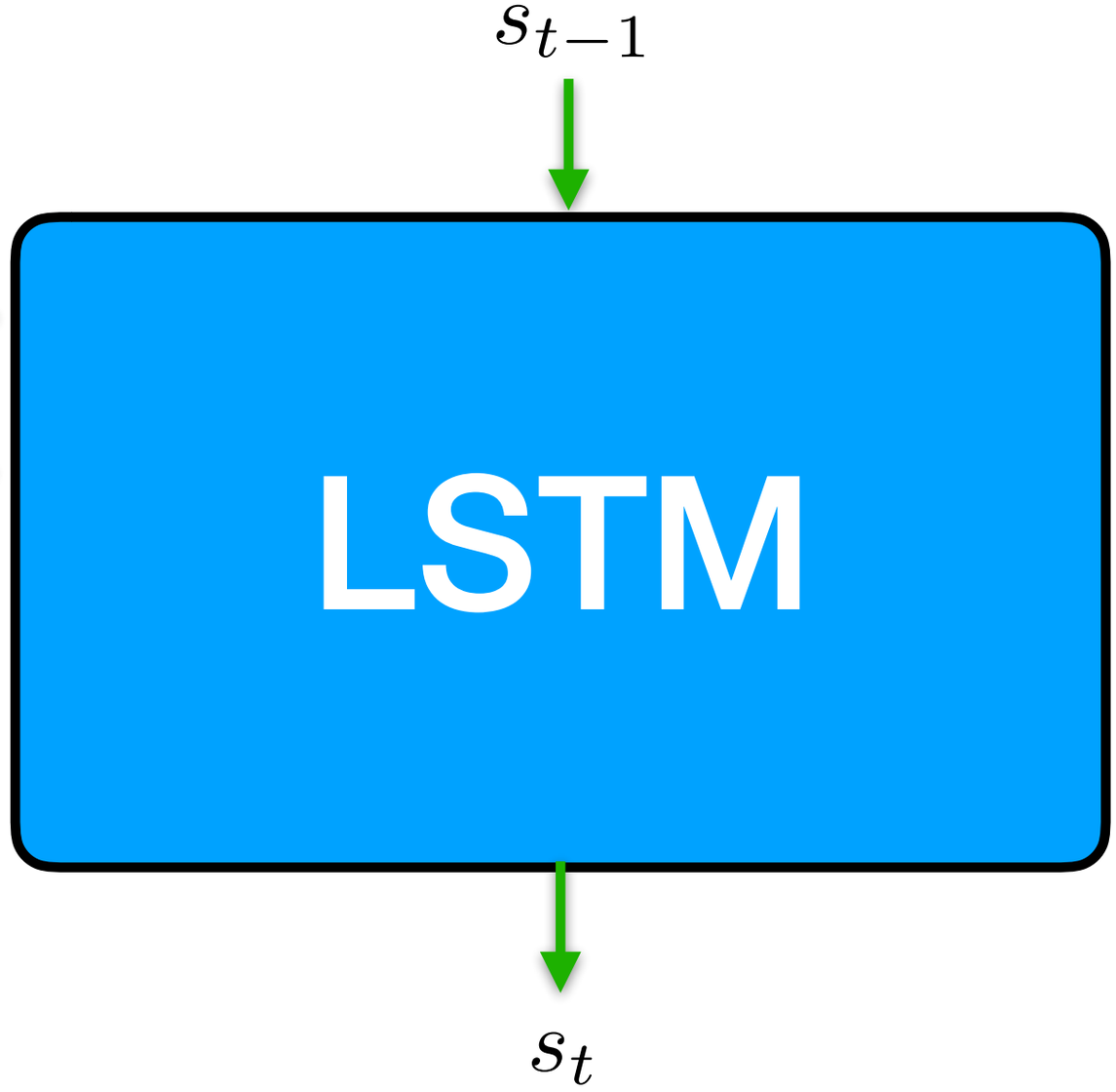
Iyer et al., 2018. Yin et al., 2017.

Decoder RNN



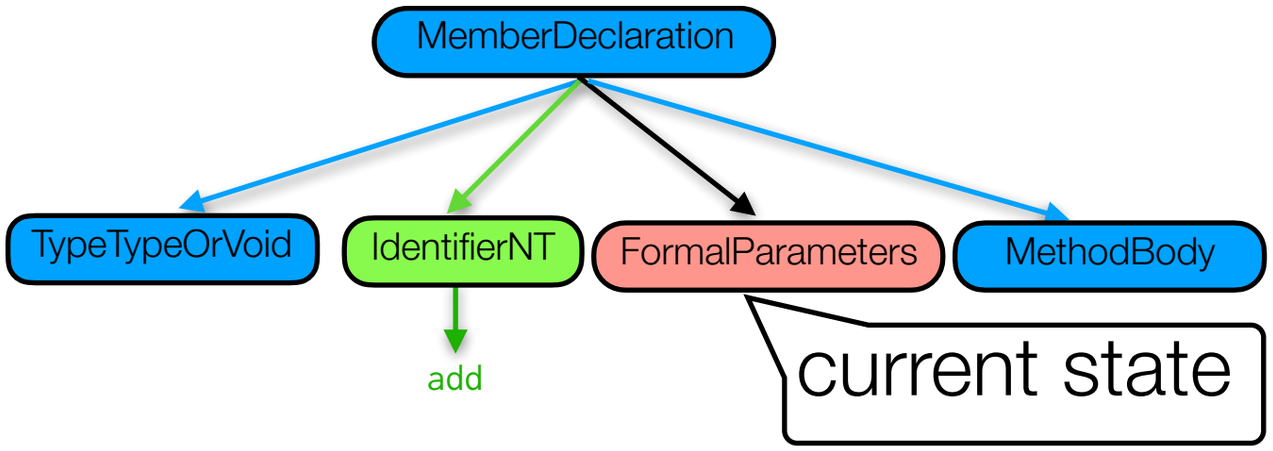
FormalParameters (n_t)

IdentifierNT -> add (a_{t-1})



Iyer et al., 2018. Yin et al., 2017.

Decoder RNN

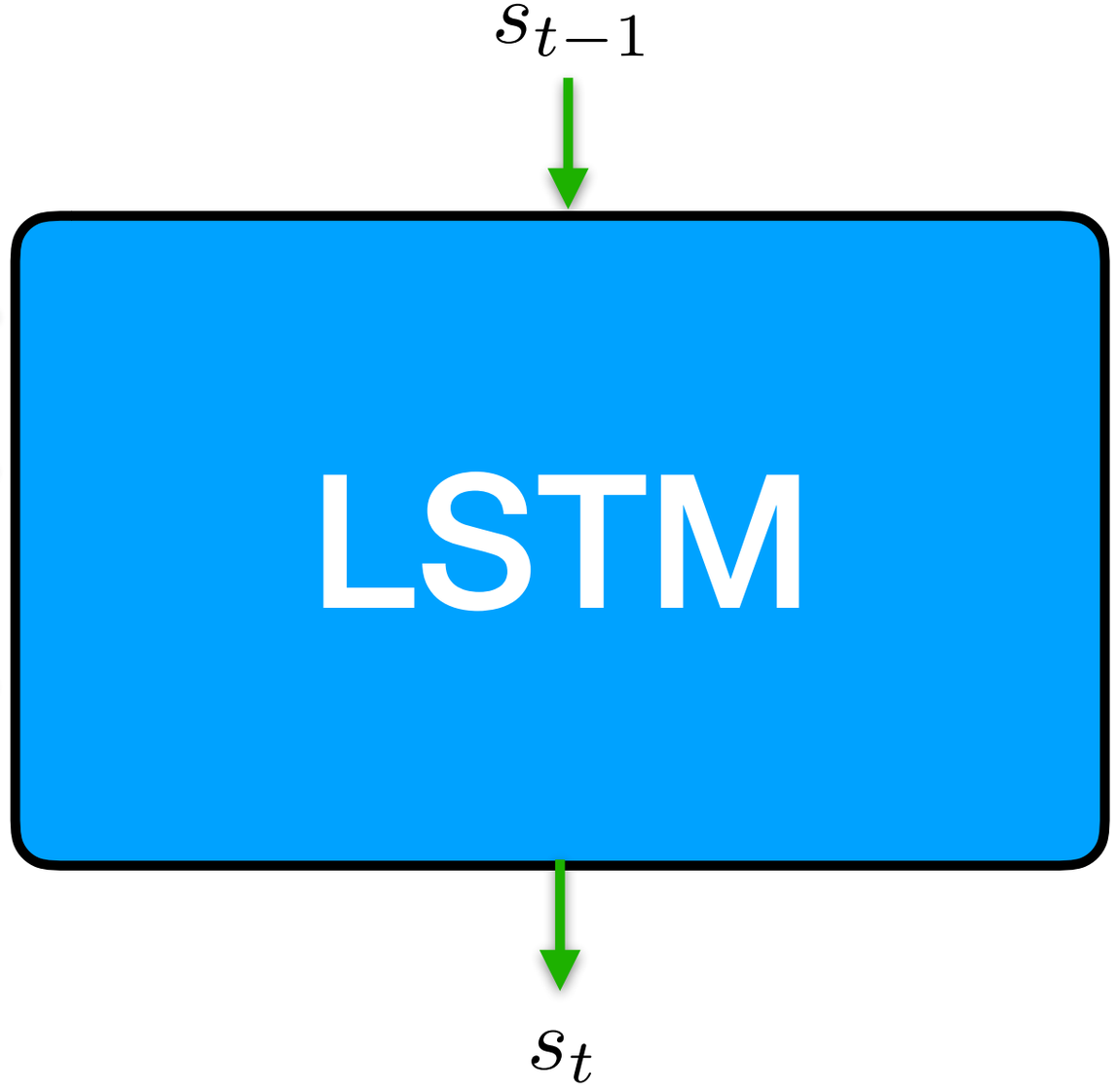


FormalParameters (n_t)

IdentifierNT -> add (a_{t-1})

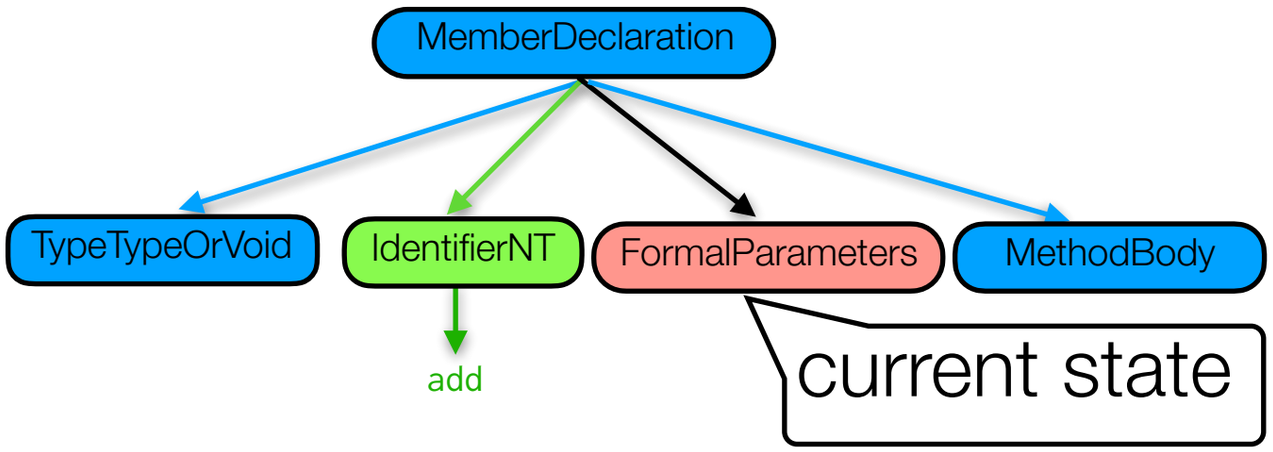
MemberDeclaration ->
TypeTypeOrVoid $(par(n_t))$
IdentifierNT
FormalParameters
MethodBody

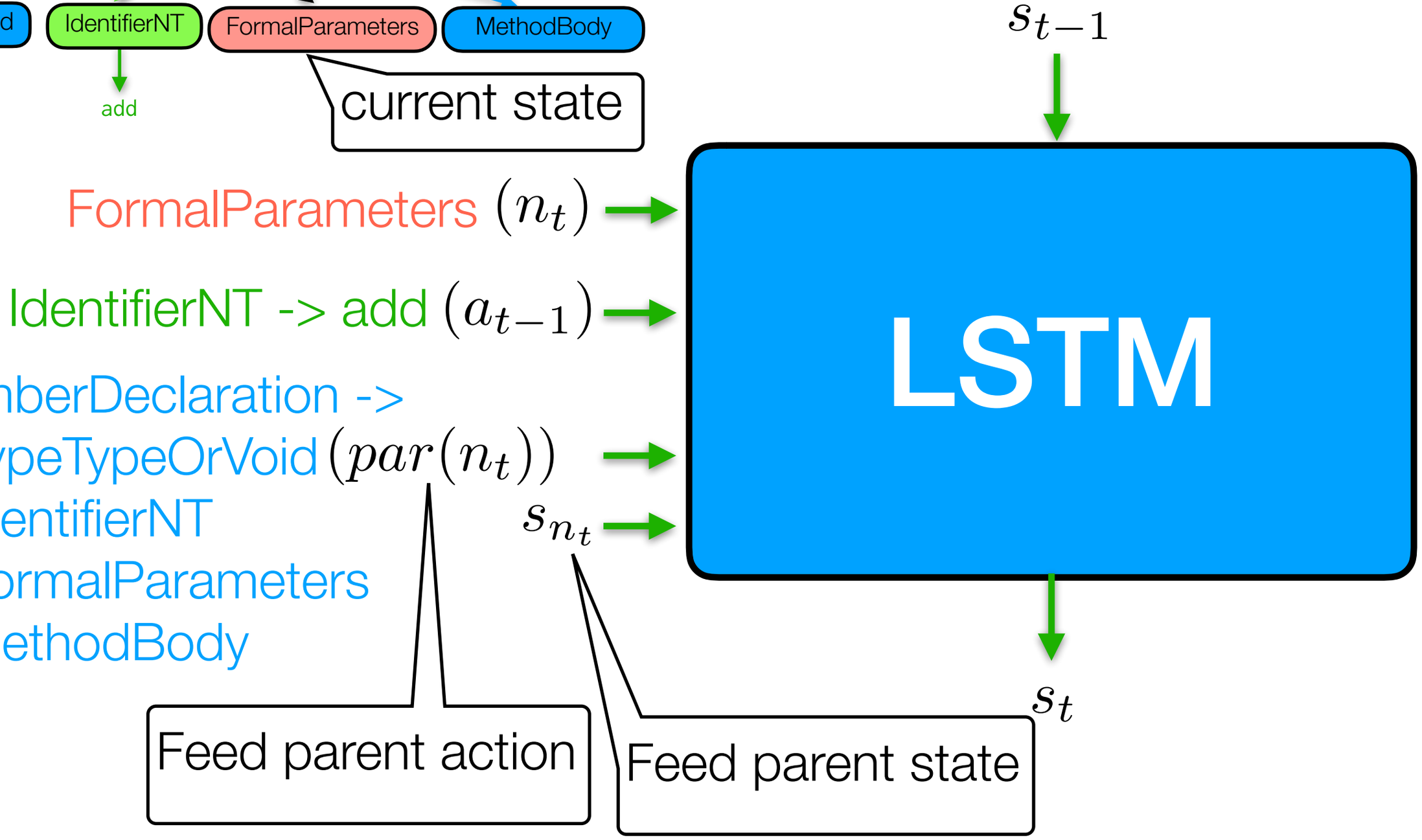
Feed parent action



Iyer et al., 2018. Yin et al., 2017.

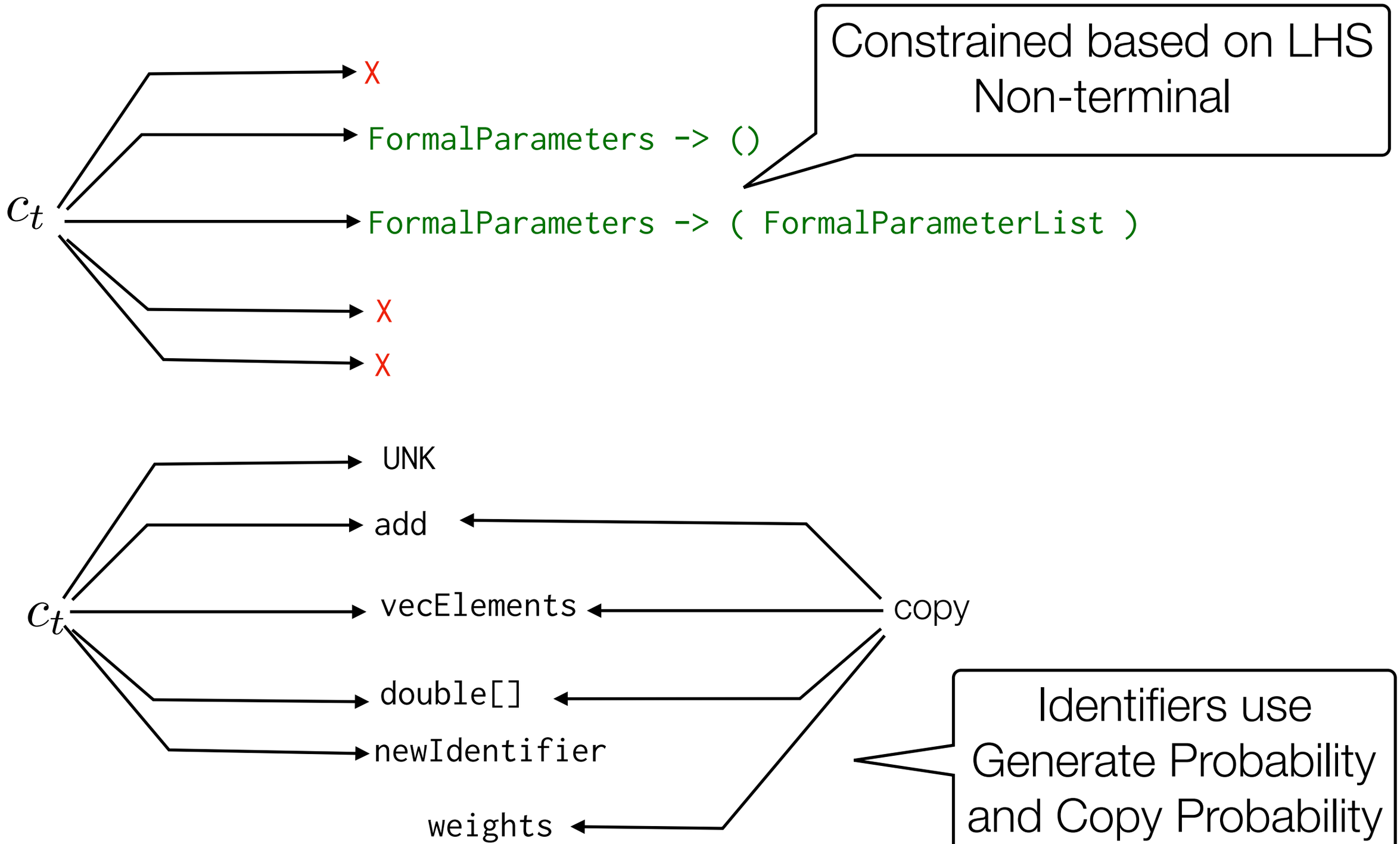
Decoder RNN





Iyer et al., 2018. Yin et al., 2017.

Output Generation



Long chains of Production Rules

```
TypeTypeOrVoid --> TypeType
TypeType --> Nt_41
Nt_41 --> PrimitiveType
PrimitiveType --> double
```

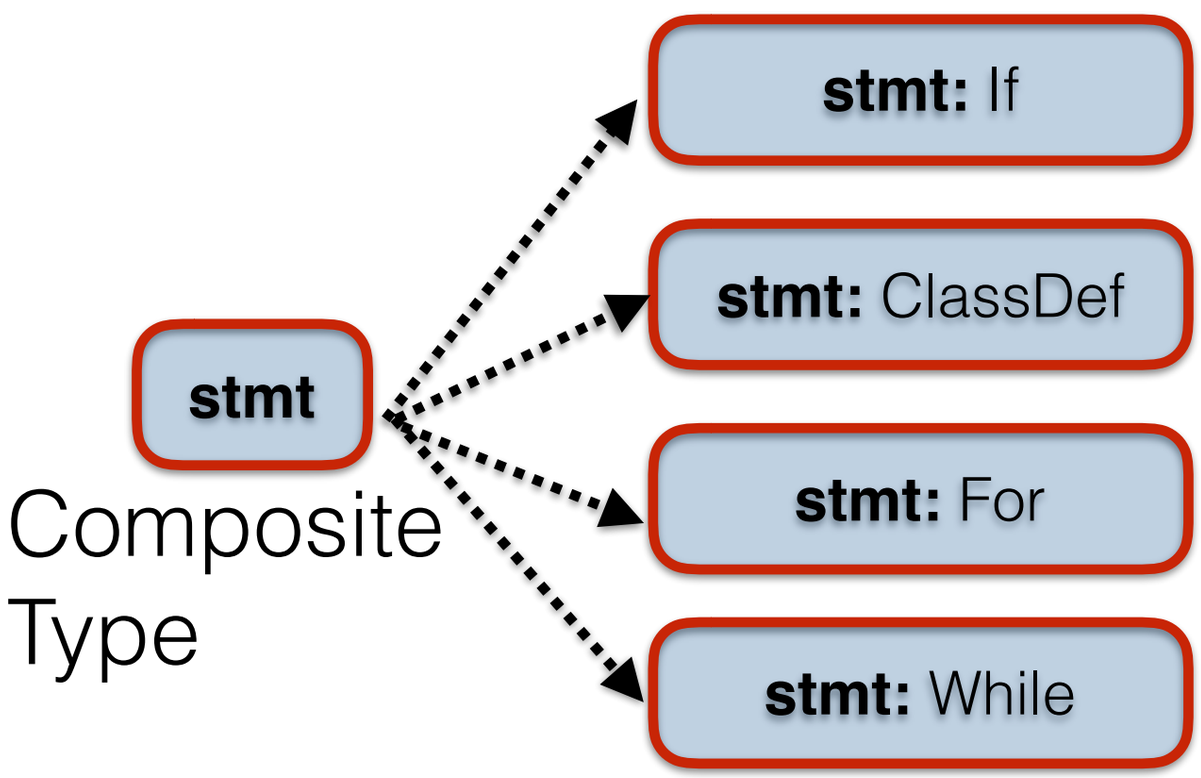
Unary Closure: Replace chains of Unary Productions by a single Production

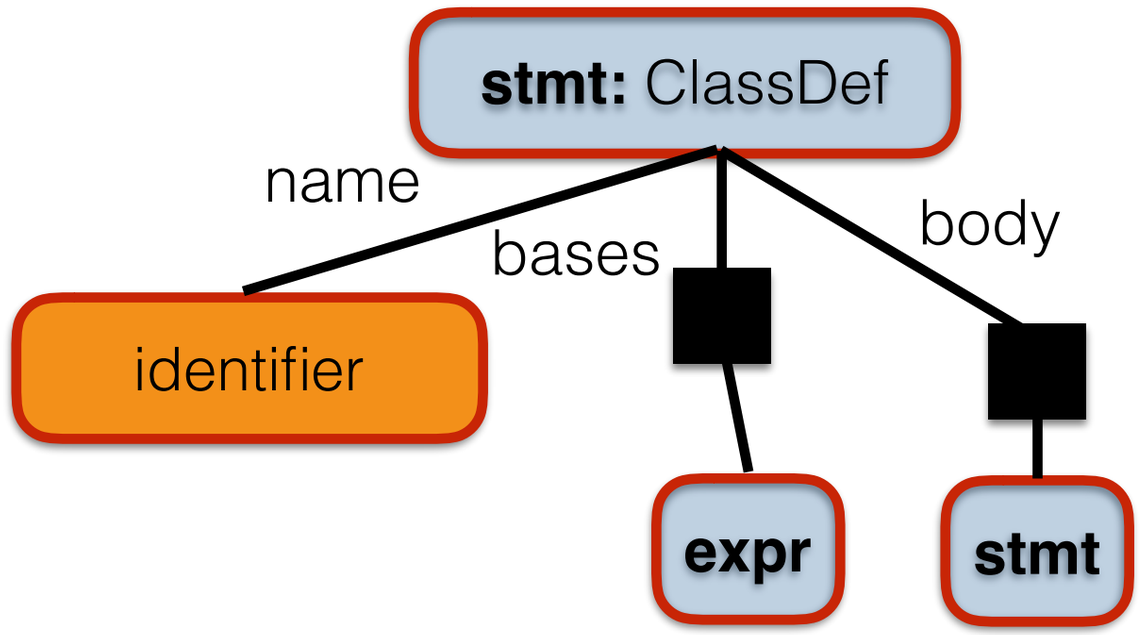
```
TypeTypeOrVoid --> TypeType | double
```

Decoding

- Token-Sequence based Decoder (Ling et al., 2016)
  - Can generate syntactically ill-formed code
- Production Rules based Decoder (Yin et al., 2017, Iyer et al., 2018)
- Abstract Syntax Networks (Rabinovich et al., 2017)

Abstract Syntax Networks





Composite Nodes have children that have a **name** and a **type**

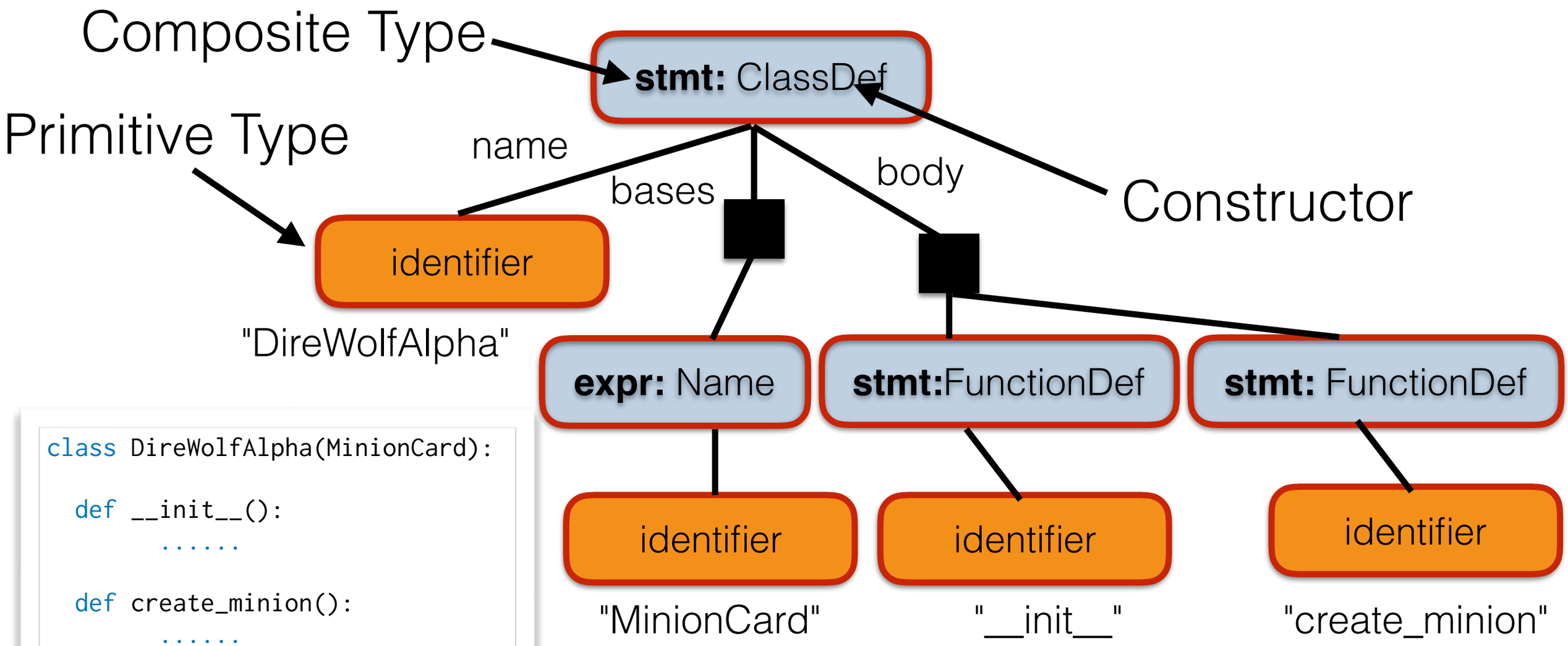
Primitive Type

"DireWolfAlpha"

Abstract Syntax Description Language

Rabinovich et al., 2017.

Abstract Syntax Networks

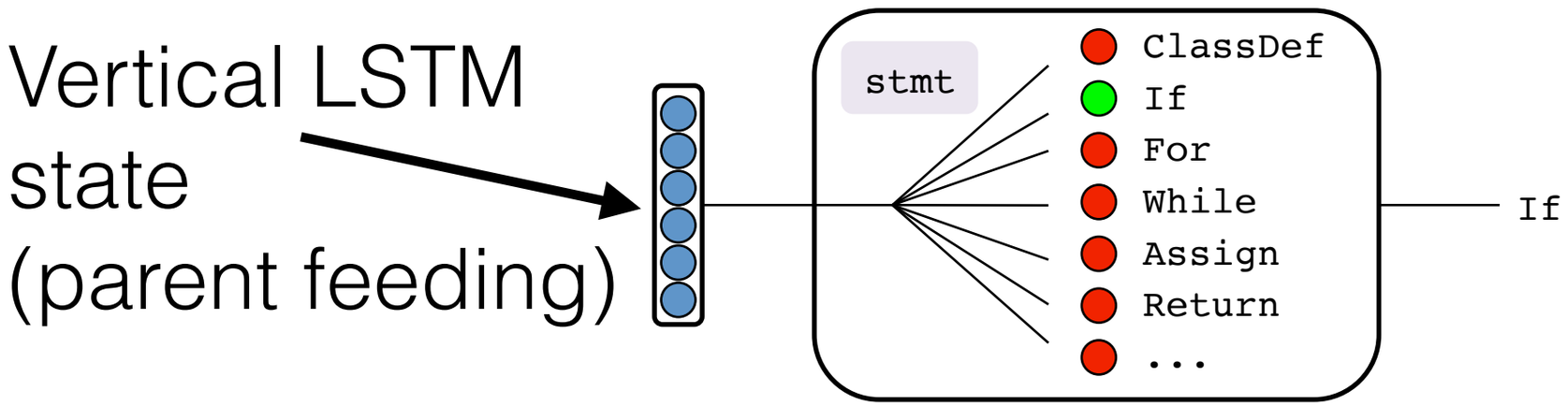



```
class DireWolfAlpha(MinionCard):

  def __init__():
        ......

  def create_minion():
        ......
```

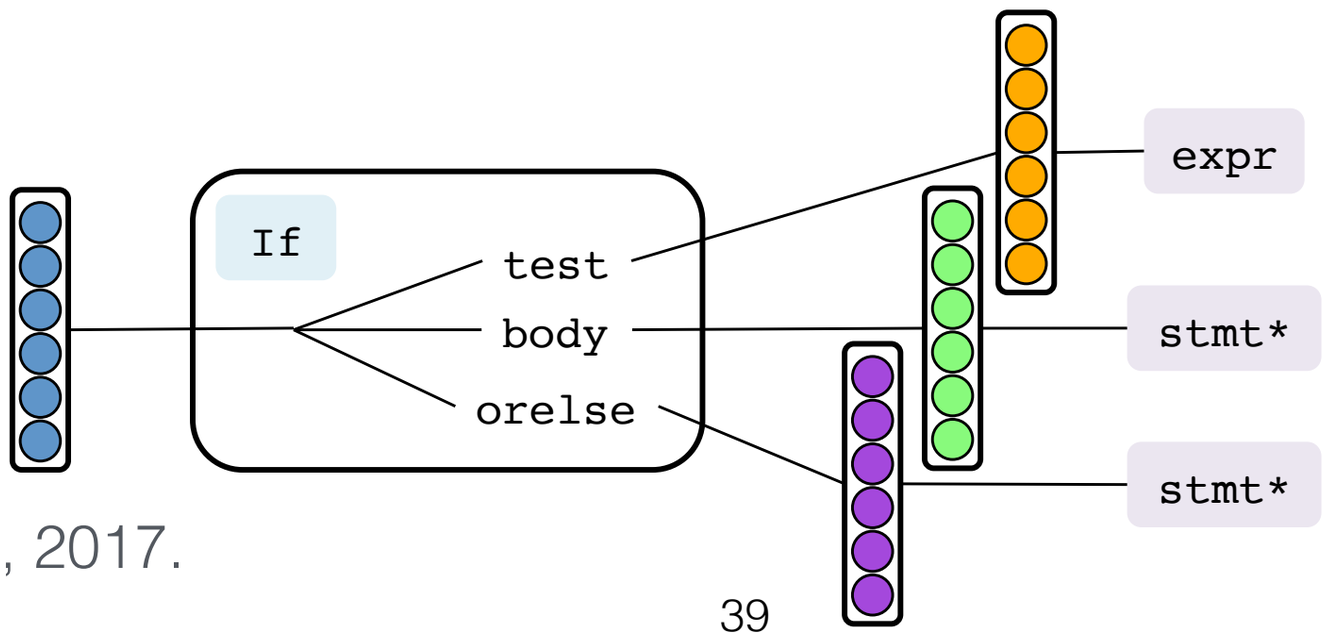
Rabinovich et al., 2017.

Mutually Recursive Decoding

Every Composite Type has a Decoder Module.



Every Constructor has a Decoder Module.

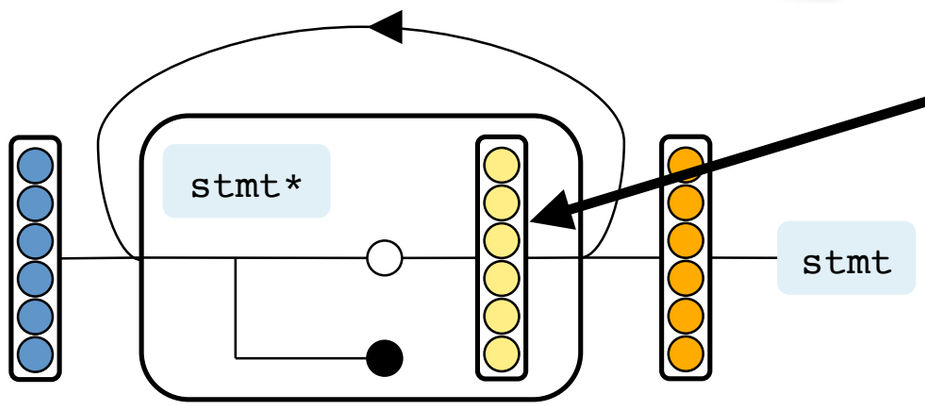


Rabinovich et al., 2017.

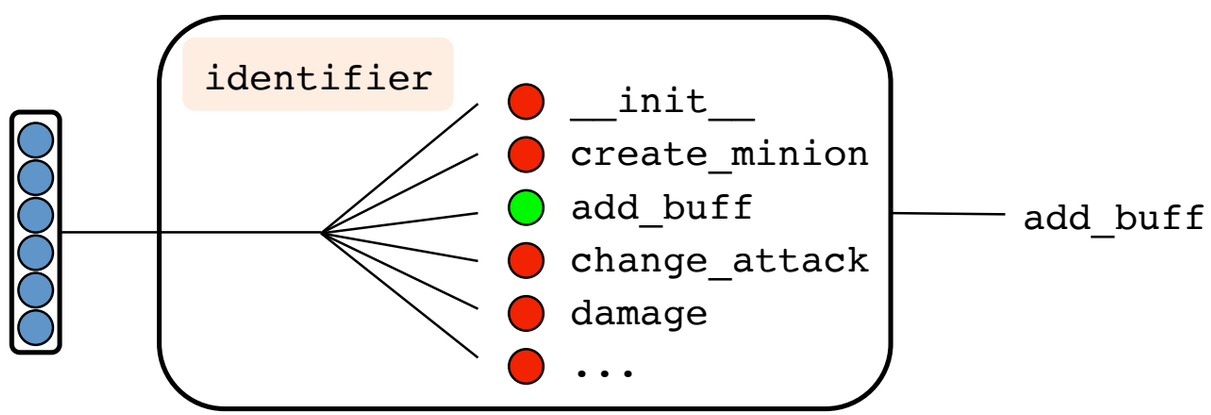
Mutually Recursive Decoding

Fields with cardinality > 1

Horizontal LSTM state



Decoder Modules for Primitive Types



Rabinovich et al., 2017.

Copy Mechanisms


```
class DivineFavor(SpellCard):
  def __init__(self):
    super().__init__(
      "Divine Favor",
      3,
      CHARACTER_CLASS.PALADIN,
      CARD_RARITY.RARE)

  def use(self, player, game):
    super().use(player, game)
    difference =
      len(game.other_player.hand) - len(player.hand)
    for i in range(0, difference):
      player.draw()
```

Copy Mechanisms


```
class DivineFavor(SpellCard):
  def __init__(self):
    super().__init__(
      "Divine Favor",
      3,
      CHARACTER_CLASS.PALADIN,
      CARD_RARITY.RARE)

  def use(self, player, game):
    super().use(player, game)
    difference =
      len(game.other_player.hand) - len(player.hand)
    for i in range(0, difference):
      player.draw()
```

Copy Mechanisms


```
class DivineFavor(SpellCard):
  def __init__(self):
    super().__init__(
      "Divine Favor",
      3,
      CHARACTER_CLASS.PALADIN,
      CARD_RARITY.RARE)

  def use(self, player, game):
    super().use(player, game)
    difference =
      len(game.other_player.hand) - len(player.hand)
    for i in range(0, difference):
      player.draw()
```

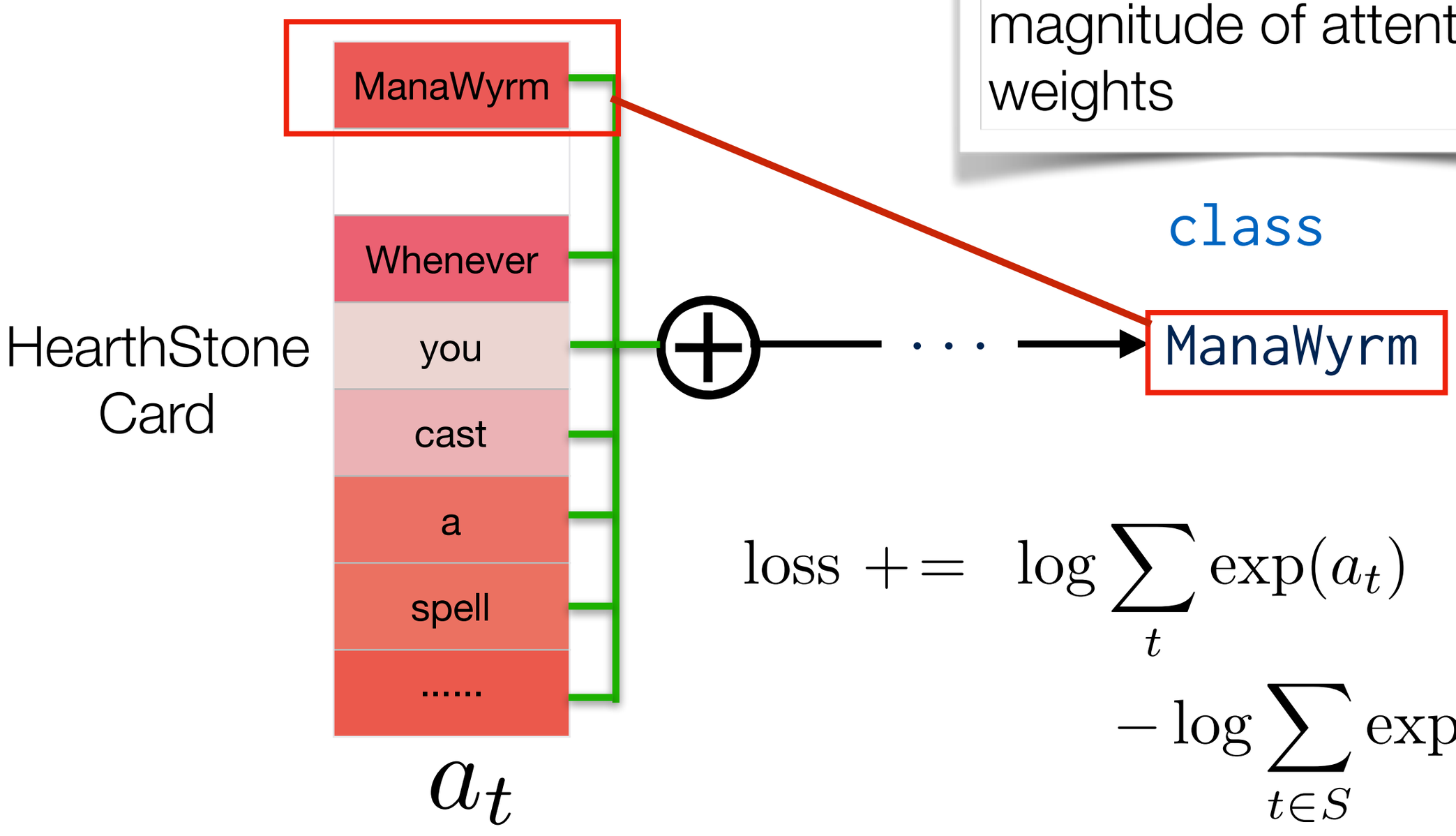
Copy Mechanisms


```
class DivineFavor(SpellCard):
  def __init__(self):
    super().__init__(
      "Divine Favor",
      3,
      CHARACTER_CLASS.PALADIN,
      CARD_RARITY.RARE)

  def use(self, player, game):
    super().use(player, game)
    difference =
      len(game.other_player.hand) - len(player.hand)
    for i in range(0, difference):
      player.draw()
```

- Supervised Attention
- Explicit Copy Mechanism

Supervised Attention

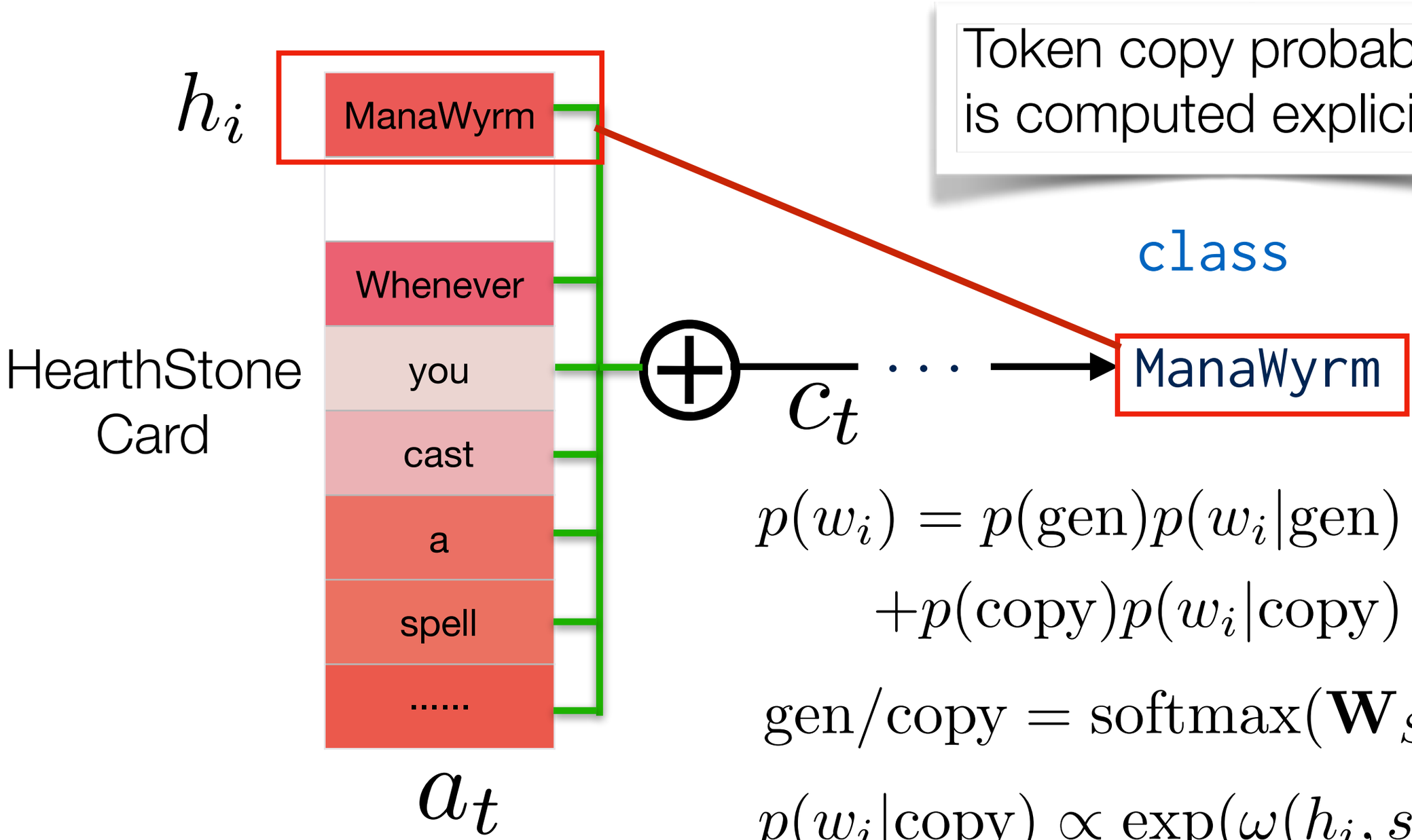
Tokens copied based on magnitude of attention weights



$$\text{loss} \mathrel{+}= \log \sum_t \exp(a_t) - \log \sum_{t \in S} \exp(a_t)$$

Rabinovich et al., 2017. Abstract Syntax Networks for Code Generation.

Explicit Copy Mechanism (Gu et al., 2016)



$$p(w_i) = p(\text{gen})p(w_i|\text{gen}) + p(\text{copy})p(w_i|\text{copy})$$

$$\text{gen/copy} = \text{softmax}(\mathbf{W}_S \cdot s_t)$$

$$p(w_i|\text{copy}) \propto \exp(\omega(h_i, s_t, c_t))$$

Ling et al., 2016, Yin et al., 2017, Lin et al., 2018, Iyer et al., 2018

Evaluation Metrics

- Exact Match
- Partial Credit using BLEU Score
- F1 score on AST Nodes
- Unit Tests

Evaluation Metrics

- Exact Match
- Partial Credit using BLEU Score
- F1 score on AST Nodes
- Unit Tests

Every single token should exactly match!

No human post editing required!

System	Accuracy
DJANGO	71.6 (Yin et al., 2017)
Hearthstone	22.7 (Rabinovich et al.)
CONCODE	8.3 (Iyer et al., 2018)

Evaluation Metrics

- Exact Match
- Partial Credit using BLEU Score
- F1 score on AST Nodes
- Unit Tests

Evaluation Metrics

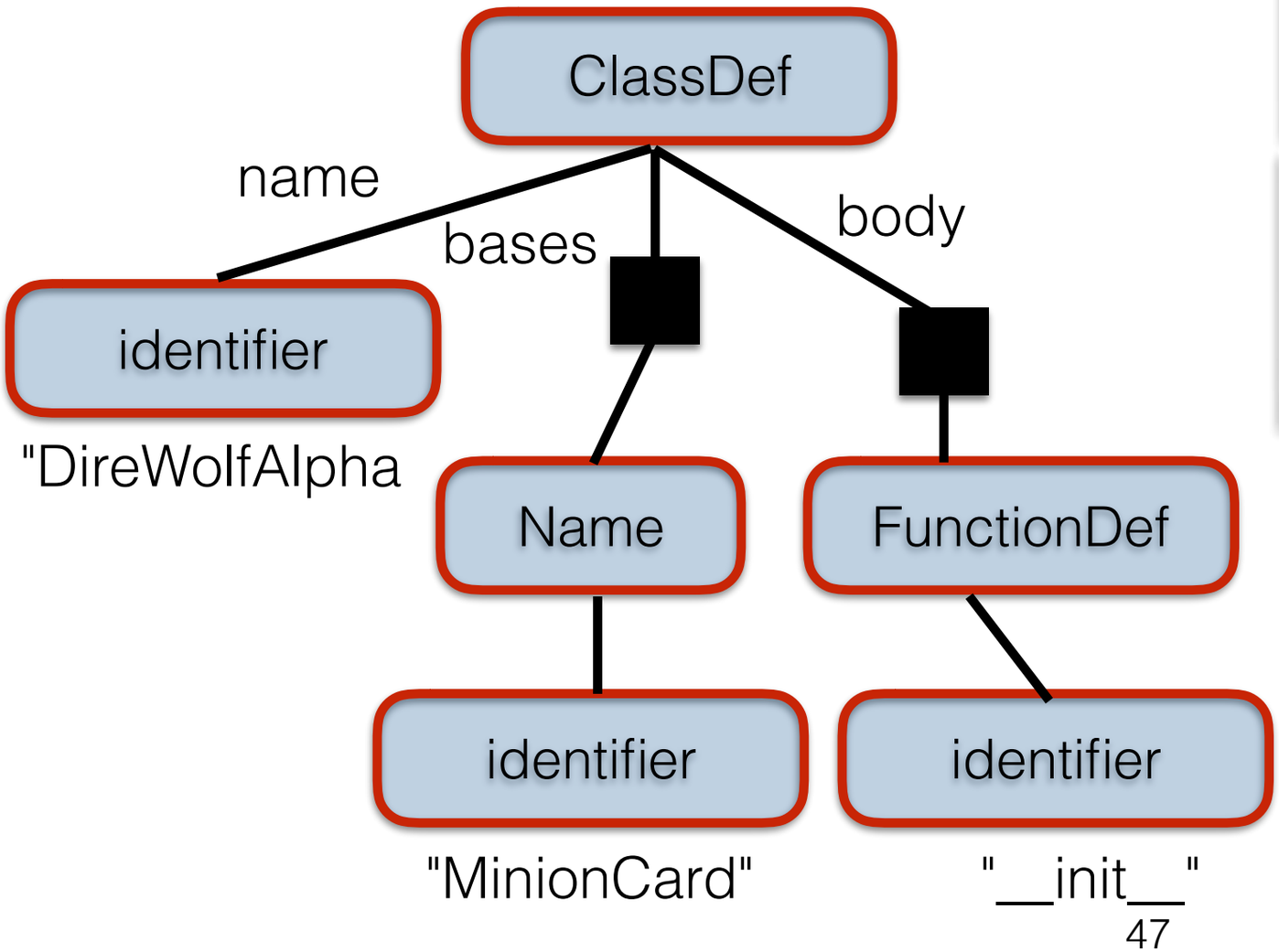
- Exact Match
- Partial Credit using BLEU Score
- F1 score on AST Nodes
- Unit Tests

BLEU=64.2

```
class Preparation(SpellCard):
  def __init__(self):
    super().__init__("Preparation", 0,
    CHARACTER_CLASS.ROGUE, CARD_RARITY.EPIC,
    target_func=hearthbreaker.targeting.
      find_minion_spell_target)

  def use(self, player, game):
    super().use(player, game)
    self.target.change_attack(3)
    player.add_aura(AuraUntil(ManaChange(-3))
      CardSelector(condition=IsSpell()), SpellCast()))
```

Evaluation Metrics

- Exact Match
- Partial Credit using BLEU Score
- F1 score on AST Nodes
- Unit Tests



Use largest common tree prefix to compute intersection

$$\text{Precision} = \frac{\#\text{intersection}}{\#\text{prediction}}$$

$$\text{Recall} = \frac{\#\text{intersection}}{\#\text{gold}}$$

Evaluation Metrics

- Exact Match
- Partial Credit using BLEU Score
- F1 score on AST Nodes
- Unit Tests

Iyer et al., 2018

Variables:

```
long numPptEntries
List<Violation> violations
```

Empty the violations list

Prediction:

```
void function() {
  violations.clear() ;
}
```

Reference:

```
void function () {
  violations = new
      ArrayList<Violation>();
}
```

Evaluation Metrics

- Exact Match
- Partial Credit using BLEU Score
- F1 score on AST Nodes
- Unit Tests

Obtain unit tests from Maven/Ant test cases in GitHub repositories


```
@Test
void testVectorAdd() {
  Vector a = new Vector(Arrays.asList(1, 1, 1));
  a.add(5);
  Vector b = new Vector(Arrays.asList(6, 6, 6));
  assertTrue(a.equals(b));
}
```

Testing the add() method

Evaluation Metrics

- Exact Match
- Partial Credit using BLEU Score
- F1 score on AST Nodes
- Unit Tests

Obtain unit tests from Maven/Ant test cases in GitHub repositories

1. Enables execution of generated source code
2. Inexpensive way to obtain test cases
3. Takes care of code dependencies

Summary

- Motivations for direct code generation
- Three granularities for generating source code
- Encoder components and methods
- Production rule based decoding
- Decoding using Abstract Syntax Networks
- Evaluation Metrics