The Minimalist Machine

Some Persian Examples

Trevor Sullivan
Joint work with Sandiway Fong

April 8, 2016

University of Arizona

Table of contents

Introduction to Minimalism

Minimalism

The diverse grammars for different languages are considered too complicated to be learnable in the very short amount of time that children learn them.

The hypothesis of Universal Grammar, the idea that the core of grammatical systems exists in human brains independent of particular language, demands a better solution.

In the 80s and 90s there came a push to reconcile all of the wildly different grammars that exist for various languages.

Minimize the complexity of grammar to a small number of systems that are plausibly innate

Minimalism

- Throwing out:

Minimalism

- Throwing out:
  - X-bar

Minimalism

- Throwing out:
  - X-bar
  - Government + Binding

Minimalism

- Throwing out:
  - X-bar
  - Government + Binding
- New concepts:

Minimalism

- Throwing out:
  - X-bar
  - Government + Binding
- New concepts:
  - Bare Phrase Structure

Minimalism

- Throwing out:
  - X-bar
  - Government + Binding
- New concepts:
  - Bare Phrase Structure
  - Phases CP and sometimes vP
    (Nothing can move out of a phase unless it is at the left edge of that phase)

Minimalism

- Throwing out:
  - X-bar
  - Government + Binding
- New concepts:
  - Bare Phrase Structure
  - Phases CP and sometimes vP
    (Nothing can move out of a phase unless it is at the left edge of that phase)
  - Interpretation through valuation of features

Feature Valuation

Interpretable Features Information that is contained within a head and can be read by other heads.
eg: plurality in "cats" and tense on "did"

Uninterpretable Features Features present on a head that have no information of their own *until* they are matched with an interpretable feature. uFs are deleted at the Conceptual-Intensional interface, if valued.

Edge Features Feature that is satisfied when an item merges into the head's left edge (spec) position. There is debate over whether Edge Features are their own category, or a subtype of uF.

Uninterpretable feature and edge feature valuation is performed by an action called "agree" through a "probe-goal" mechanism that reaches down into the tree.

Theory note

The minimalist program is not without critics or faults.

Theory note

The minimalist program is not without critics or faults.

1. No empirical evidence.
   There is no scientific evidence that motivates the existence of the minimalist program, it is not more descriptively powerful than traditional grammars, but it is more theoretically "perfect".

Theory note

The minimalist program is not without critics or faults.

1. No empirical evidence.
   There is no scientific evidence that motivates the existence of the minimalist program, it is not more descriptively powerful than traditional grammars, but it is more theoretically "perfect".
2. Very vague.
   Most discussions of the Minimalist Program do not go deep into the details. For most writers, the question of whether it could work in theory is more important than an investigation of whether it does.

The Minimalist Machine

The Machine

The minimalist machine is a program that constructs derivations according to the minimalist program.[1]

[1] Based primarily on Chomsky(2001): Derivation by Phase

The Machine

The minimalist machine is a program that constructs derivations according to the minimalist program.[1]

In other words, it builds sentences in the same way that brains do.*

[1] Based primarily on Chomsky(2001): Derivation by Phase

The Machine

The minimalist machine is a program that constructs derivations according to the minimalist program.[1]

In other words, it builds sentences in the same way that brains do.*

An attempt to prove the viability of the Minimalist Program as the core component of UG. It is a mathematical model of what the human language faculty has to compute.

[1] Based primarily on Chomsky(2001): Derivation by Phase

How does it work?

- Prolog

How does it work?

- Prolog
- Two parts:

How does it work?

- Prolog
- Two parts:
  1. Universal Grammar

How does it work?

- Prolog
- Two parts:
  1. Universal Grammar
     - Merge

How does it work?

- Prolog
- Two parts:
  1. Universal Grammar
     - Merge
     - Agree

How does it work?

- Prolog
- Two parts:
  1. Universal Grammar
     - Merge
     - Agree
  2. Language-Specific Grammar

How does it work?

- Prolog
- Two parts:
  1. Universal Grammar
     - Merge
     - Agree
  2. Language-Specific Grammar
     - Lexicon

How does it work?

- Prolog
- Two parts:
  1. Universal Grammar
     - Merge
     - Agree
  2. Language-Specific Grammar
     - Lexicon
     - Features

Derivation Process

Derivation Process

Input Stream

- List of heads in reverse order of their merge.

Stack

Syntactic Object (SO)

Derivation Process

Input Stream

- List of heads in reverse order of their merge.

Stack

- Stack (CS Data Structure) of constituents with unvalued features.

Syntactic Object (SO)

Derivation Process

Input Stream

- List of heads in reverse order of their merge.

Stack

- Stack (CS Data Structure) of constituents with unvalued features.
- Representative of probe-goal.

Syntactic Object (SO)

Derivation Process

Input Stream

- List of heads in reverse order of their merge.

Stack

- Stack (CS Data Structure) of constituents with unvalued features.
- Representative of probe-goal.

Syntactic Object (SO)

- Built up over time by merging the existing SO with stream heads or stack constituents.

Post Derivation

Post Derivation

1. Morpheme Realization

Post Derivation

1. Morpheme Realization
   - Each head in the derivation is spelled out according to rules set in the language-specific grammar.

Post Derivation

1. Morpheme Realization
   - Each head in the derivation is spelled out according to rules set in the language-specific grammar.
2. Affix Hopping

Post Derivation

1. Morpheme Realization
   - Each head in the derivation is spelled out according to rules set in the language-specific grammar.
2. Affix Hopping
   - Suffixes hop over the word immediately to their right.

Post Derivation

1. Morpheme Realization
   - Each head in the derivation is spelled out according to rules set in the language-specific grammar.
2. Affix Hopping
   - Suffixes hop over the word immediately to their right.
3. Morpheme Realization part 2

Post Derivation

1. Morpheme Realization
   - Each head in the derivation is spelled out according to rules set in the language-specific grammar.
2. Affix Hopping
   - Suffixes hop over the word immediately to their right.
3. Morpheme Realization part 2
   - Affixes join with heads.

Persian 1:01

Persian 1:01

Persian is an Iranian language spoken in Iran, Afghanistan, and Tajikistan.

Persian 1:01

Persian is an Iranian language spoken in Iran, Afghanistan, and Tajikistan.

The written dialect/style is different from the spoken dialect/style. The difference is primarily in pronunciation, which does not matter for us, but where there is a grammatical difference, the spoken form is preferred.

Persian 1:01

Persian is an Iranian language spoken in Iran, Afghanistan, and Tajikistan.

The written dialect/style is different from the spoken dialect/style. The difference is primarily in pronunciation, which does not matter for us, but where there is a grammatical difference, the spoken form is preferred.

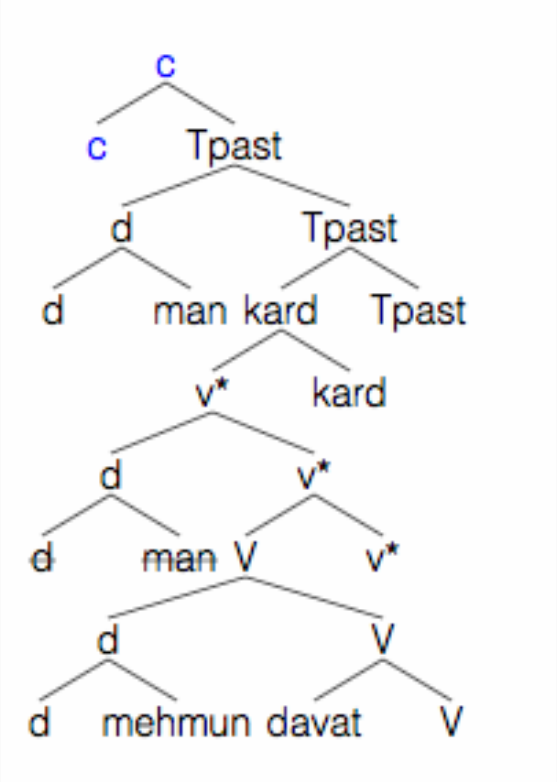
SOV word order...

Persian 1:01

SOV word order...

Unlike most SOV languages, Persian has prepositions rather than to postpositions, determiners that precede nouns, and complementizers before their related clauses.

Therefore, we can assert with confidence, as many have, that Persian is a left-headed language in all parameters except the verbal structure.



Persian Challenges to the Minimalist Program

Complex Predicate

In Persian, many verbal concepts are expressed not through verbs, but rather through the combination of a non-verb and a semantically bleached light verb. These are henceforth referred to as “NVE” and “LV”.

Complex Predicate

In Persian, many verbal concepts are expressed not through verbs, but rather through the combination of a non-verb and a semantically bleached light verb. These are henceforth referred to as "NVE" and "LV".

(3) man mehmun davat kardam
1.SG guest invitation do.PST-1.SG

'I invited a guest'

lit. 'I did invitation a guest'

(4) Bahar zabun yâd gereft-Ø
B. language memory take.PST-3.SG

'Bahar learned a language'

lit. 'Bahar took memory a language'

Complex Predicate

In Persian, many verbal concepts are expressed not through verbs, but rather through the combination of a non-verb and a semantically bleached light verb. These are henceforth referred to as "NVE" and "LV".

(5) man mehmun davat kardam
1.SG guest invitation do.PST-1.SG
'I invited a guest'
lit. 'I did invitation a guest'

(6) Bahar zabun yâd gereft-Ø
B. language memory take.PST-3.SG
'Bahar learned a language'
lit. 'Bahar took memory a language'

Complex Predicates are often idiomatic in meaning, but not in nature: they take an object, can be separated by adverbs (sometimes), can be transitive or intransitive (ie, the NVE is *not* a direct object).

Complex Predicate

Solution:

Complex Predicate

Solution: Build in a pre-narrow-syntax morphology.

Complex Predicate

Solution: Build in a pre-narrow-syntax morphology.

Add to the Persian Grammar module a verbalizer. The verbalizer attaches to a non-verbal head, and the resulting SO is *not* a constituent, but rather, remains a head.

Complex Predicate

Solution: Build in a pre-narrow-syntax morphology.

Add to the Persian Grammar module a verbalizer. The verbalizer attaches to a non-verbal head, and the resulting SO is *not* a constituent, but rather, remains a head.

Step 3.

Begin substream

Stream:	[[davat!D],[V!N]]
Stack (⇓):	
SO:	[]

Complex Predicate

Solution: Build in a pre-narrow-syntax morphology.

Add to the Persian Grammar module a verbalizer. The verbalizer attaches to a non-verbal head, and the resulting SO is *not* a constituent, but rather, remains a head.

Step 4.

Head of stream [davat!D] is the initial SO

Stream:	[[V!N]]
Stack (⇓):	
SO:	[davat!D] davat!D

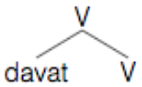
Complex Predicate

Solution: Build in a pre-narrow-syntax morphology.

Add to the Persian Grammar module a verbalizer. The verbalizer attaches to a non-verbal head, and the resulting SO is *not* a constituent, but rather, remains a head.

Step 5.

Merge [V!N] and [davat!D]
Label from [V!N] (syntactic head with an unvalued uF)
[V!N] values D on [davat!D]
[davat] values N on [V!N]

Stream:	[]
Stack (⇓):	
SO:	[V[davat][V]] 

End substream, SO [V[davat][V]] re-inserted into main stream

Complex Predicate

Solution: Build in a pre-narrow-syntax morphology.

Add to the Persian Grammar module a verbalizer. The verbalizer attaches to a non-verbal head, and the resulting SO is *not* a constituent, but rather, remains a head.

Step 6.

Stream:	[[V[davat][V]],[v*!phi],[[man!D],[d!case!N]],[kard],[Tpast!phi],[c]]
Stack (⇓):	
SO:	[d!case[d][mehmun]] d!case d mehmun

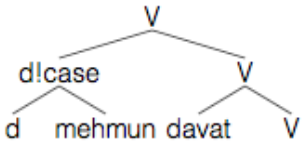
Complex Predicate

Solution: Build in a pre-narrow-syntax morphology.

Add to the Persian Grammar module a verbalizer. The verbalizer attaches to a non-verbal head, and the resulting SO is *not* a constituent, but rather, remains a head.

Step 7.

Merge [V[davat][V]] and [d!case[d][mehmun]]
Label from [V[davat][V]] (syntactic head merging with a non-head)
Theta-mark [d!case[d][mehmun]]
Push [d!case[d][mehmun]] (unvalued uF) onto stack

Stream:	[[v*!phi],[[man!D],[d!case!N]],[kard],[Tpast!phi],[c]]
Stack (⇓):	[d!case[d][mehmun]]
SO:	[V[d!case[d][mehmun]][V[davat][V]]] 

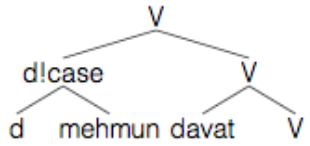
Complex Predicate

Solution: Build in a pre-narrow-syntax morphology.

Add to the Persian Grammar module a verbalizer. The verbalizer attaches to a non-verbal head, and the resulting SO is *not* a constituent, but rather, remains a head.

Step 7.

Merge [V[davat][V]] and [d!case[d][mehmun]]
Label from [V[davat][V]] (syntactic head merging with a non-head)
Theta-mark [d!case[d][mehmun]]
Push [d!case[d][mehmun]] (unvalued uF) onto stack

Stream:	[[v*!phi],[[man!D],[d!case!N]],[kard],[Tpast!phi],[c]]
Stack (⇓):	[d!case[d][mehmun]]
SO:	[V[d!case[d][mehmun]][V[davat][V]]] 

This kind of operation is not precedented as part of narrow syntax, but syntacticians often assume that certain morphological operations can take place before narrow syntax.

Complex Predicate

What about LV?

Complex Predicate

What about LV?

Scholarly consensus is that the light verb in Complex Predicate is a v. That is uncontested here.

Many LVs are possible, more than the traditional count of little vs, and they serve some semantic purpose, whereas v is purely functional, eg

(11) yâd gereftan
memory taking
'learn'

is semantically distinct from

(12) yâd âvardan
memory getting
'remember'

whereas

(13) (v*) eat steak
. .

has the same semantic content as

(14) steak (v̆) eaten
. .

v can move arguments, but not change the core semantics of a word.

Complex Predicate

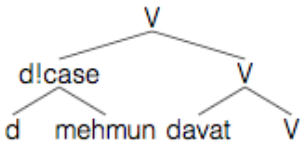
Therefore, we consider v to be distinct from LV, and as a result, the LV is merged to the tree at its own level.

Complex Predicate

Therefore, we consider v to be distinct from LV, and as a result, the LV is merged to the tree at its own level.

Step 7.

Merge [V[davat][V]] and [d!case[d][mehmun]]
Label from [V[davat][V]] (syntactic head merging with a non-head)
Theta-mark [d!case[d][mehmun]]
Push [d!case[d][mehmun]] (unvalued uF) onto stack

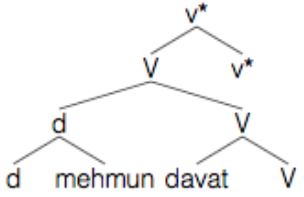
Stream:	[[v*!phi],[[man!D],[d!case!N]],[kard],[Tpast!phi],[c]]
Stack (⇓):	[d!case[d][mehmun]]
SO:	[V[d!case[d][mehmun]][V[davat][V]]] 

Complex Predicate

Therefore, we consider v to be distinct from LV, and as a result, the LV is merged to the tree at its own level.

Step 8.

Merge [v*!phi] and [V[d!case[d..][mehmun..]][V[davat..][V..]]]
Label from [v*!phi] (syntactic head with an unvalued uF)
[d!case[d][mehmun]] values uPhi on [v*!phi]
[v*] values acc case on [d!case[d][mehmun]]

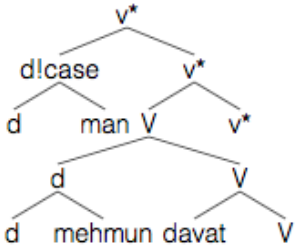
Stream:	[[[man!D],[d!case!N]],[kard],[Tpast!phi],[c]]
Stack (⇓):	[d[d][mehmun]]
SO:	[v*[V[d[d][mehmun]][V[davat][V]]][v*]] 

Complex Predicate

Therefore, we consider v to be distinct from LV, and as a result, the LV is merged to the tree at its own level.

Step 13.

Merge [v*[V[d..][V..]][v*]] and [d!case[d][man]]
Label from [v*[V[d..][V..]][v*]] (edge feature)
Theta-mark [d!case[d][man]]
Push [d!case[d][man]] (unvalued uF) onto stack

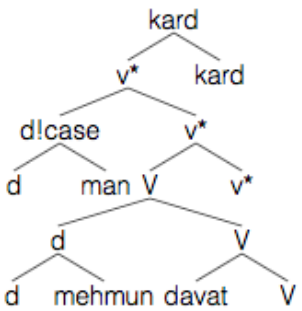
Stream:	[[kard],[Tpast!phi],[c]]
Stack (⇓):	[d!case[d][man]] [d[d][mehmun]]
SO:	[v*[d!case[d][man]][v*[V[d[d][mehmun]][V[davat][V]]][v*]]] 

Complex Predicate

Therefore, we consider v to be distinct from LV, and as a result, the LV is merged to the tree at its own level.

Step 14.

Merge [kard] and [v*[d!case[d..][man..]][v*[V..][v*..]]]
Label from [kard] (syntactic head merging with a non-head)

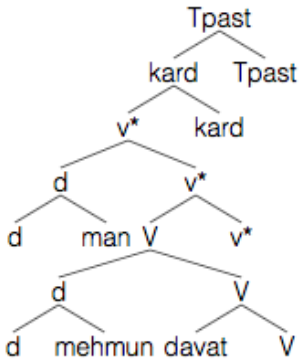
Stream:	[[Tpast!phi],[c]]
Stack (⇓):	[d!case[d][man]] [d[d][mehmun]]
SO:	[kard[v*[d!case[d][man]][v*[V[d[d][mehmun]][V[davat][V]]][v*]]][kard]] 

Complex Predicate

Therefore, we consider v to be distinct from LV, and as a result, the LV is merged to the tree at its own level.

Step 15.

Merge [Tpast!phi] and [kard[v*[d..][v*..]][kard]]
Label from [Tpast!phi] (syntactic head merging with a non-head)
[d!case[d][man]] values uPhi on [Tpast!phi]
[Tpast] values nom case on [d!case[d][man]]

Stream:	[[c]]
Stack (⇓):	[d[d][man]] [d[d][mehmun]]
SO:	[Tpast[kard[v*[d[d][man]][v*[V[d[d][mehmun]][V[davat][V]]][v*]]][kard]][Tpast]] 

Verbal Morphology

Verbal Morphology

Verbal Morphology

Since T is right-headed (as it is part of the verbal complex, this is the null hypothesis), it is trivial to get the morphology containing the subject's ϕ-features and the sentential tense in the right position, but there are three verbal morphemes that are difficult to place in the tree structure. Each of the following are prefixes that attach to the main verb, or light verb in Complex Predicates.

Verbal prefixes		
mi- (durative aspect)	na- (negation)	be- (subjunctive)

Verbal Morphology

Since T is right-headed (as it is part of the verbal complex, this is the null hypothesis), it is trivial to get the morphology containing the subject's ϕ-features and the sentential tense in the right position, but there are three verbal morphemes that are difficult to place in the tree structure. Each of the following are prefixes that attach to the main verb, or light verb in Complex Predicates.

Verbal prefixes		
mi- (durative aspect)	na- (negation)	be- (subjunctive)

These things appear to represent verbal concepts, and as such we would expect them to be right-headed like the rest of the verbal complex. However, this would put them on the wrong side of the verb to which they attach.

Verbal Morphology

Since T is right-headed (as it is part of the verbal complex, this is the null hypothesis), it is trivial to get the morphology containing the subject's ϕ-features and the sentential tense in the right position, but there are three verbal morphemes that are difficult to place in the tree structure. Each of the following are prefixes that attach to the main verb, or light verb in Complex Predicates.

Verbal prefixes		
mi- (durative aspect)	na- (negation)	be- (subjunctive)

These things appear to represent verbal concepts, and as such we would expect them to be right-headed like the rest of the verbal complex. However, this would put them on the wrong side of the verb to which they attach.

It has been proposed that they are left-headed, but this would require motivating absolutely everything under them, includind NVE to move left past it.

A new definition for Affix Hopping

A new definition for Affix Hopping

This problem can be solved by making a new definition for affix hopping.

Head-Initial Phrases	Head-Final Phrases
▪ Prefixes do not hop. ▪ Suffixes hop.	▪ Prefixes hop. ▪ Suffixes do not hop.

A new definition for Affix Hopping

This problem can be solved by making a new definition for affix hopping.

Head-Initial Phrases	Head-Final Phrases
▪ Prefixes do not hop. ▪ Suffixes hop.	▪ Prefixes hop. ▪ Suffixes do not hop.

This may not be the most theoretically pleasing, since some of the structure from narrow syntax is leaking into the phonological operations post-syntax, but it functions too beautifully to set aside. Furthermore, it makes a testable prediction, something that Minimalism has in short supply.

A new definition for Affix Hopping

Head-Initial Phrases	Head-Final Phrases
▪ Prefixes do not hop. ▪ Suffixes hop.	▪ Prefixes hop. ▪ Suffixes do not hop.

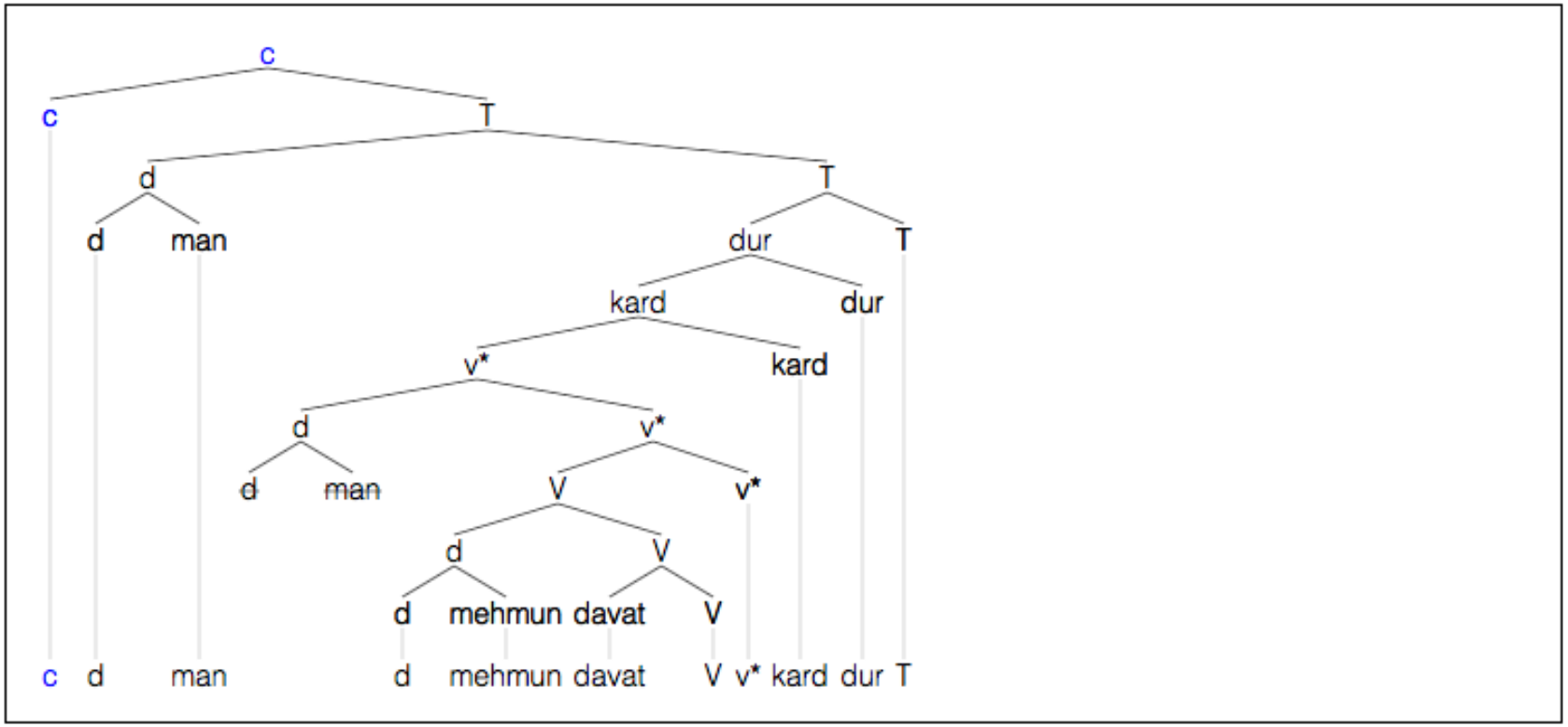
This may not be the most theoretically pleasing, since some of the structure from narrow syntax is leaking into the phonological operations post-syntax, but it functions too beautifully to set aside. Furthermore, it makes a testable prediction, something that Minimalism has in short supply.

This is realized in the Machine with a different type of hyphen that the head in question spells out with.

Head-Initial Phrases	Head-Final Phrases
x~ prefix that does not hop -x - suffix that hops	x= - prefix that hops #x - suffix that does not hop

Affix Hopping Example

Parse:



Spell-out:
man -acc mehmun davat kard mi= t(pres,[1,sg]) (after morpheme realization)
man mehmun -acc davat mi= kard t(pres,[1,sg]) (after affix-hop)
man mehmun -acc davat mi= kon #am (after morpheme realization, stage 2)
man mehmun davat mi-kon-am

Specific Object Movement

Specific Objects

Persian has a specificity distinction, but only in direct objects. It is marked with the phrase-level suffix "-ro" ("-râ" in written and formal speech)

The difference is not only in interpretation, however, because the specific and nonspecific objects have different distributions.

(15) Bahar diruz miz tamiz kard-Ø
B. yesterday table clean do.PST-3.SG
'Bahar cleaned the table yesterday'

(16) Bahar miz-a-ro diruz tamiz kard-Ø
B. table-PL-SPF yesterday clean do.PST-3.SG
'Bahar cleaned the (particular) tables yesterday'

Specific Objects

Furthermore, there is a stark contrast concerning which item can be deleted with the NVE in a complex predicate.

Specific Objects

Furthermore, there is a stark contrast concerning which item can be deleted with the NVE in a complex predicate.

(18) *Bahar miz tamiz kard-Ø, vali Reza pangere kard-Ø
B. table clean do.PST-3.SG but R. window do.PST-3.SG

'Bahar cleaned the table, but Reza did the window'

Specific Objects

Furthermore, there is a stark contrast concerning which item can be deleted with the NVE in a complex predicate.

(19) *Bahar miz tamiz kard-Ø, vali Reza pangere kard-Ø
B. table clean do.PST-3.SG but R. window do.PST-3.SG

'Bahar cleaned the table, but Reza did the window'

(20) Bahar miz-a-ro tamiz kard-Ø, vali Reza pangere-ha-ro kard-Ø
B. table-PL-SPF clean do.PST-3.SG but R. window-PL-SPF do.PST-3.SG

'Bahar cleaned the (particular) tables, but Reza did the (particular) windows'

Specific Objects

Furthermore, there is a stark contrast concerning which item can be deleted with the NVE in a complex predicate.

(21) *Bahar miz tamiz kard-Ø, vali Reza pangere kard-Ø
B. table clean do.PST-3.SG but R. window do.PST-3.SG

'Bahar cleaned the table, but Reza did the window'

(22) Bahar miz-a-ro tamiz kard-Ø, vali pangere-ha-ro na-kard-Ø
B. table-PL-SPF clean do.PST-3.SG but window-PL-SPF NEG-do.PST-3.SG

'Bahar cleaned the (particular) tables, but didn't the (particular) windows'

Proposed Explanation

[2] Karimi, Mahdavi, Nabors, Smith, Sullivan (2016)

Proposed Explanation

If it is assumed that prepositional phrases attach to V, the first examples merely require the specific object to move left of V, with no regards to where.

[2] Karimi,Mahdavi,Nabors,Smith,Sullivan(2016)

Proposed Explanation

If it is assumed that prepositional phrases attach to V, the first examples merely require the specific object to move left of V, with no regards to where.

The second set of ellipsis examples constrains the movement somewhat. At first glance, one is tempted to say that the specific object moves out of vP (or VoiP, if you prefer) [2], but this presents a problem in minimalism. Firstly, (if one desires the LV to be v or Voi), it requires head movement, which is not allowed in a minimalist framework, but aside from that, it breaks the probe-goal chain.

[2] Karimi,Mahdavi,Nabors,Smith,Sullivan(2016)

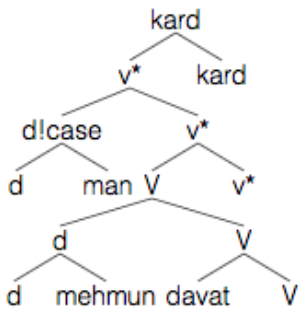
Proposed Explanation

The second set of ellipsis examples constrains the movement somewhat. At first glance, one is tempted to say that the specific object moves out of vP (or VoiP, if you prefer) [2], but this presents a problem in minimalism. Firstly, (if one desires the LV to be v or Voi), it requires head movement, which is not allowed in a minimalist framework, but aside from that, it breaks the probe-goal chain.

Step 14.

Merge [kard] and [v*[d!case[d..][man..]][v*[V..][v*..]]]
Label from [kard] (syntactic head merging with a non-head)

Stream:	[[Tpast!phi],[c]]
Stack (⇓):	[d!case[d][man]] [d[d][mehmun]]
SO:	[kard[v*[d!case[d][man]][v*[V[d[d][mehmun]][V[davat][V]]][v*]]][kard]]



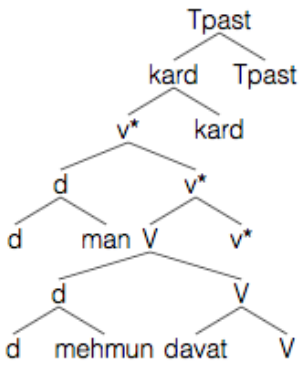
[2] Karimi,Mahdavi,Nabors,Smith,Sullivan(2016)

Proposed Explanation

The second set of ellipsis examples constrains the movement somewhat. At first glance, one is tempted to say that the specific object moves out of vP (or VoiP, if you prefer) [2], but this presents a problem in minimalism. Firstly, (if one desires the LV to be v or Voi), it requires head movement, which is not allowed in a minimalist framework, but aside from that, it breaks the probe-goal chain.

Step 15.

Merge [Tpast!phi] and [kard[v*[d..][v*..]][kard]]
Label from [Tpast!phi] (syntactic head merging with a non-head)
[d!case[d][man]] values uPhi on [Tpast!phi]
[Tpast] values nom case on [d!case[d][man]]

Stream:	[[c]]
Stack (⇓):	[d[d][man]] [d[d][mehmun]]
SO:	[Tpast[kard[v*[d[d][man]][v*[V[d[d][mehmun]][V[davat][V]]][v*]]][kard]][Tpast]] 

[2] Karimi Mahdavi Nabors Smith Sullivan(2016)

Proposed Explanation

[3] This has precedent in Karimi(2005)

Proposed Explanation

It is therefore proposed that the specific object moves to a position below or at v/Voi.[3]

[3] This has precedent in Karimi(2005)

Proposed Explanation

It is therefore proposed that the specific object moves to a position below or at v/Voi.[3]

The ellipsis mentioned previously is therefore VP ellipsis, not vP ellipsis.

[3] This has precedent in Karimi(2005)

Proposed Explanation

It is therefore proposed that the specific object moves to a position below or at v/Voi.[3]

The ellipsis mentioned previously is therefore VP ellipsis, not vP ellipsis.

(26) Bahar miz-a-ro tamiz na-kard-Ø, vali
B. table-PL-SPF clean NEG-do.PST-3.SG but
pangere-ha-ro kard-Ø
window-PL-SPF do.PST-3.SG

'Bahar cleaned the (particular) tables, but didn't the (particular) windows'

[3]This has precedent in Karimi(2005)

Proposed Explanation

It is therefore proposed that the specific object moves to a position below or at v/Voi.[3]

The ellipsis mentioned previously is therefore VP ellipsis, not vP ellipsis.

(27) Bahar miz-a-ro tamiz na-kard-Ø, vali
B. table-PL-SPF clean NEG-do.PST-3.SG but
pangere-ha-ro kard-Ø
window-PL-SPF do.PST-3.SG

'Bahar cleaned the (particular) tables, but didn't the (particular) windows'

The subject is a red herring

[3]This has precedent in Karimi(2005)

Proposed Explanation

It is therefore proposed that the specific object moves to a position below or at v/Voi.[3]

The ellipsis mentioned previously is therefore VP ellipsis, not vP ellipsis.

(28) Bahar miz-a-ro tamiz na-kard-Ø, vali
B. table-PL-SPF clean NEG-do.PST-3.SG but
pangere-ha-ro kard-Ø
window-PL-SPF do.PST-3.SG

'Bahar cleaned the (particular) tables, but didn't the (particular) windows'

The subject is a red herring: The observation that it can or cannot be explicitly present in the same ellipsis context implies that it is not part of the ellipsis domain, and is instead a second example NP-ellipsis that happens to co-occur with VP-ellipsis.

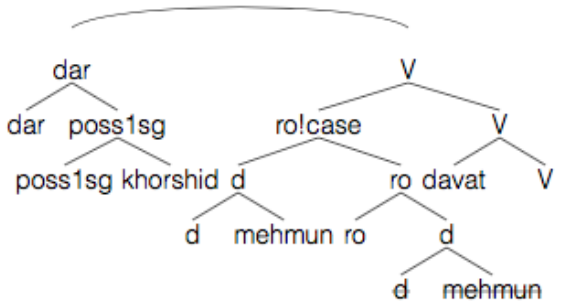
[3] This has precedent in Karimi(2005)

Specific Object Movement example

Step 15.

Pair-merge [dar[dar][poss1sg[poss1sg][khorshid]]] and [V[ro!case[d[d][mehmun]][ro[ro][d[d][mehmun]]]][V[davat][V]]]

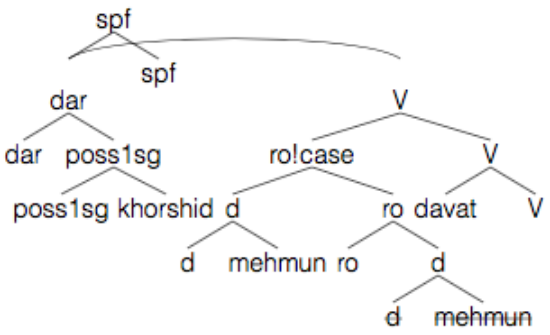
Stream:	[[spf],[v*!phi],[[man!D],[d!case!N]],[kard],[dur],[T!phi],[c]]
Stack (⇓):	[ro!case[d[d][mehmun]][ro[ro][d[d][mehmun]]]] [d[d][mehmun]]
SO:	<[V[ro!case[d[d][mehmun]][ro[ro][d[d][mehmun]]]][V[davat][V]]],[dar[dar][poss1sg[poss1sg][khorshid]]]>



Specific Object Movement example

Step 16.

Merge [spf] and <[dar[dar][poss1sg[poss1sg..][khorshid..]]],[V[ro!case[d..][ro..]][V[davat..][V..]]]>
Label from [spf] (only projecting head)

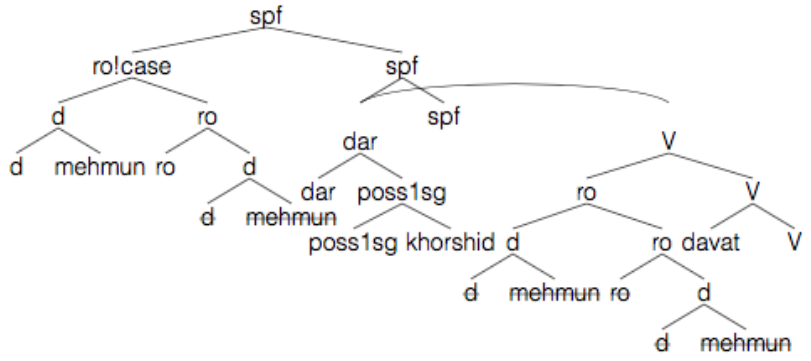
Stream:	[[v*!phi],[[man!D],[d!case!N]],[kard],[dur],[T!phi],[c]]
Stack (⇓):	[ro!case[d[d][mehmun]][ro[ro][d[d][mehmun]]]] [d[d][mehmun]]
SO:	[spf<[dar[dar][poss1sg[poss1sg][khorshid]]],[V[ro!case[d[d][mehmun]][ro[ro][d[d][mehmun]]]][V[davat][V]]]>[spf]] 

Specific Object Movement example

Step 17.

Internal merge selected
Merge [spf<[dar[dar..][poss1sg..]],[V[ro..][V..]]>[spf]] and [ro!case[d[d..][mehmun..]][ro[ro..][d..]]]
Label from [spf<[dar[dar..][poss1sg..]],[V[ro..][V..]]>[spf]] (edge feature)

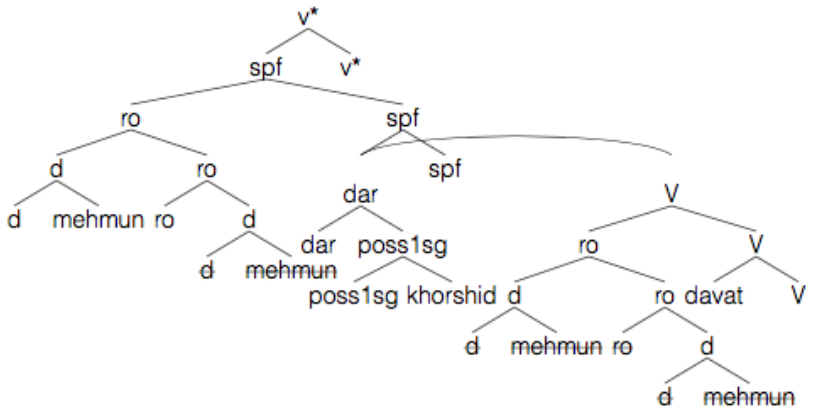
Stream:	[[v*!phi],[[man!D],[d!case!N]],[kard],[dur],[T!phi],[c]]
Stack (⇓):	[ro!case[d[d][mehmun]][ro[ro][d[d][mehmun]]]] [d[d][mehmun]]
SO:	[spf[ro!case[d[d][mehmun]][ro[ro][d[d][mehmun]]]][spf<[dar[dar][poss1sg[poss1sg][khorshid]]],[V[ro[d[d][mehmun]][ro[ro][d[d][



Specific Object Movement example

Step 18.

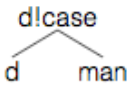
Merge [v*!phi] and [spf[ro!case[d..][ro..]][spf<[dar..],[V..]>[spf..]]]
Label from [v*!phi] (syntactic head merging with a non-head)
[ro!case[d[d..][mehmun..]][ro[ro..][d..]]] values uPhi on [v*!phi]
[v*] values acc case on [ro!case[d[d..][mehmun..]][ro[ro..][d..]]]

Stream:	[[[man!D],[d!case!N]],[kard],[dur],[T!phi],[c]]
Stack (⇓):	[ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]] [d[d][mehmun]]
SO:	[v*[spf[ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]][spf<[dar[dar][poss1sg[poss1sg][khorshid]]],[V[ro[d[d][mehmun]][ro[ro][d[d][m 

Specific Object Movement example

Step 21.

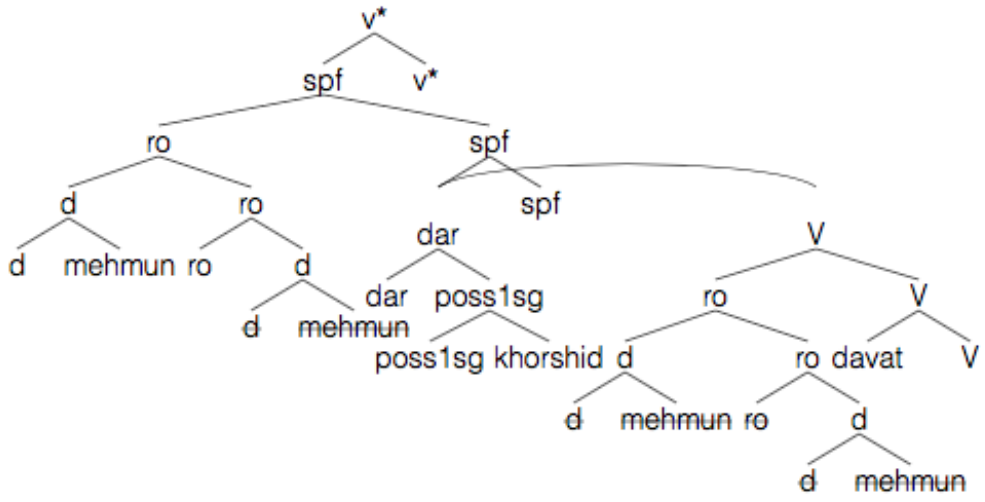
Merge [d!case!N] and [man!D]
Label from [d!case!N] (syntactic head with an unvalued uF)
Inherit interpretable feature(s) [f(phi,[1,sg,mf])] from [man!D]
[d!case!N] values D on [man!D]
[man] values N on [d!case!N]

Stream:	[]
Stack (⇓):	
SO:	[d!case[d][man]] 

End substream, SO [d!case[d][man]] re-inserted into main stream

Specific Object Movement example

Step 22.

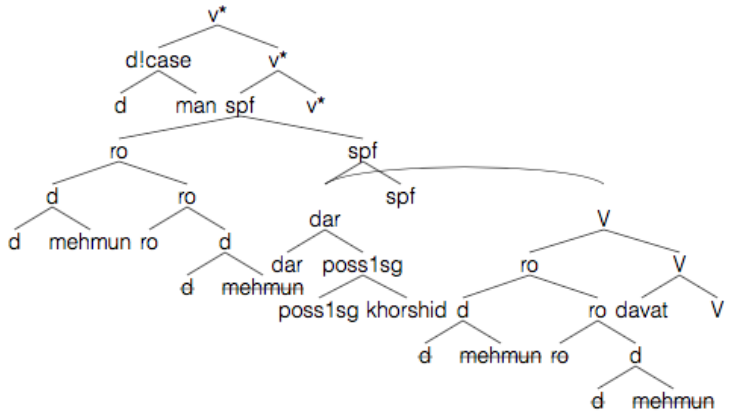
Stream:	[[d!case[d][man]],[kard],[dur],[T!phi],[c]]
Stack (⇓):	[ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]] [d[d][mehmun]]
SO:	[v*[spf[ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]][spf<[dar[dar][poss1sg[poss1sg][khorshid]]],[V[ro[d[d] 

Specific Object Movement example

Step 23.

Merge [v*[spf[ro..][spf..]][v*]] and [d!case[d][man]]
Label from [v*[spf[ro..][spf..]][v*]] (edge feature)
Theta-mark [d!case[d][man]]
Push [d!case[d][man]] (unvalued uF) onto stack

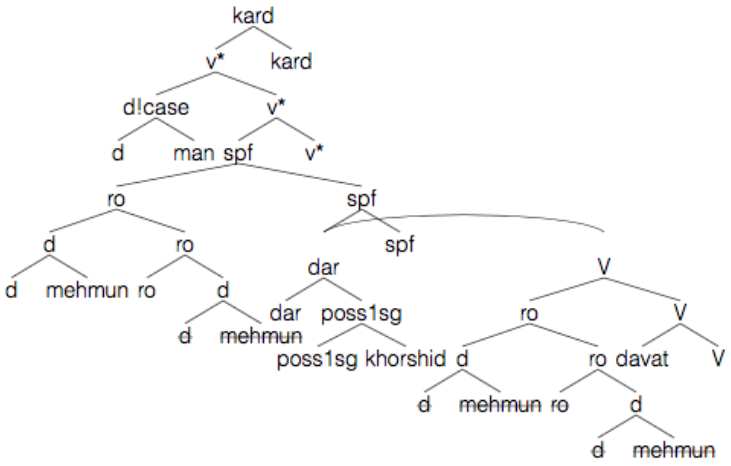
Stream:	[[kard],[dur],[T!phi],[c]]
Stack (⇓):	[d!case[d][man]] [ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]] [d[d][mehmun]]
SO:	[v*[d!case[d][man]][v*[spf[ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]][spf<[dar[dar][poss1sg[poss1sg][khorshid]]],[V[ro[d[d][mehmun]][ro[ro][d[



Specific Object Movement example

Step 24.

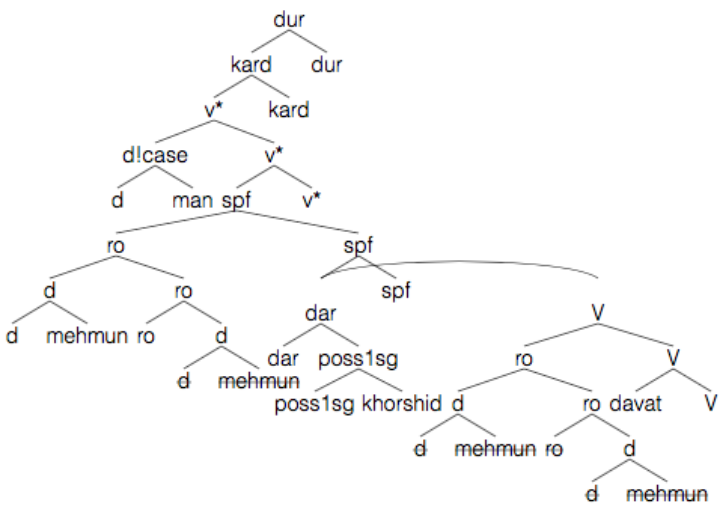
Merge [kard] and [v*[d!case[d..][man..]][v*[spf..][v*..]]]
Label from [kard] (syntactic head merging with a non-head)

Stream:	[[dur],[T!phi],[c]]
Stack (⇓):	[d!case[d][man]] [ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]] [d[d][mehmun]]
SO:	[kard[v*[d!case[d][man]][v*[spf[ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]][spf<[dar[dar][poss1sg[poss1sg][khorshid]]],[V[ro[d[d][mehmun]][ro[

Specific Object Movement example

Step 25.

Merge [dur] and [kard[v*[d..][v*..]][kard]]
Label from [dur] (syntactic head merging with a non-head)

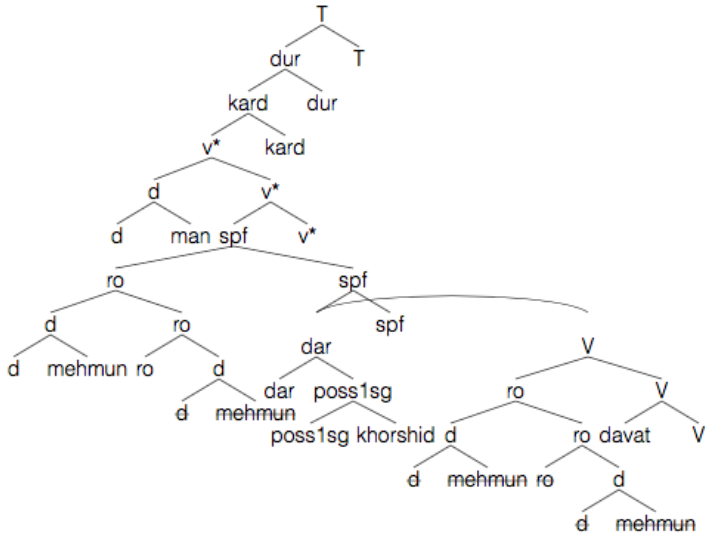
Stream:	[[T!phi],[c]]
Stack (⇓):	[d!case[d][man]] [ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]] [d[d][mehmun]]
SO:	[dur[kard[v*[d!case[d][man]][v*[spf[ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]][spf<[dar[dar][poss1sg[poss1sg][khorshid]]],[V[ro[d[d][mehmun]][r 

Specific Object Movement example

Step 26.

Merge [T!phi] and [dur[kard[v*..][kard..]][dur]]
Label from [T!phi] (syntactic head merging with a non-head)
[d!case[d][man]] values uPhi on [T!phi]
[T] values nom case on [d!case[d][man]]

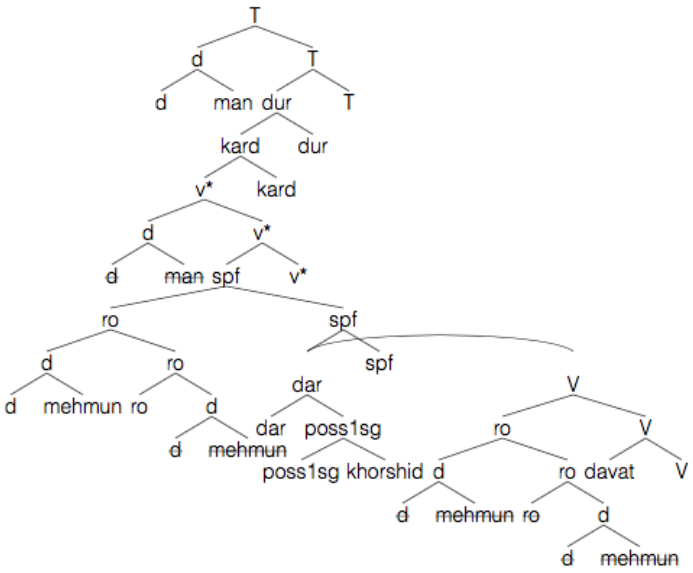
Stream:	[[c]]
Stack (⇓):	[d[d][man]] [ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]] [d[d][mehmun]]
SO:	[T[dur[kard[v*[d[d][man]][v*[spf[ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]][spf<[dar[dar][poss1sg[poss1sg][khorshid]]],[V[ro[d[d][mehmun]][ro[ro][



Specific Object Movement example

Step 27.

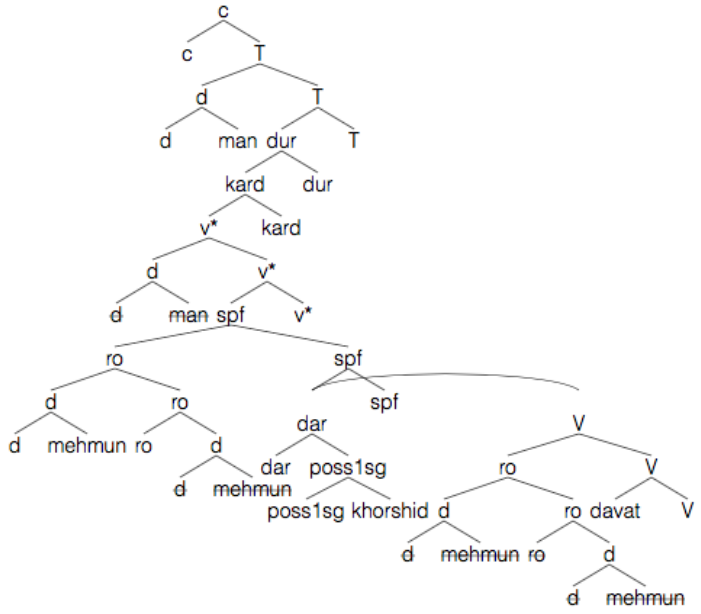
Internal merge selected
Merge [T[dur[kard..][dur..]][T]] and [d[d][man]]
Label from [T[dur[kard..][dur..]][T]] (edge feature)

Stream:	[[c]]
Stack (⇓):	[d[d][man]] [ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]] [d[d][mehmun]]
SO:	[T[d[d][man]][T[dur[kard[v*[d[d][man]][v*[spf[ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]][spf<[dar[dar][poss1sg[poss1sg][khorshid]]],[V[ro[d[d][mehmu 

Specific Object Movement example

Step 28.

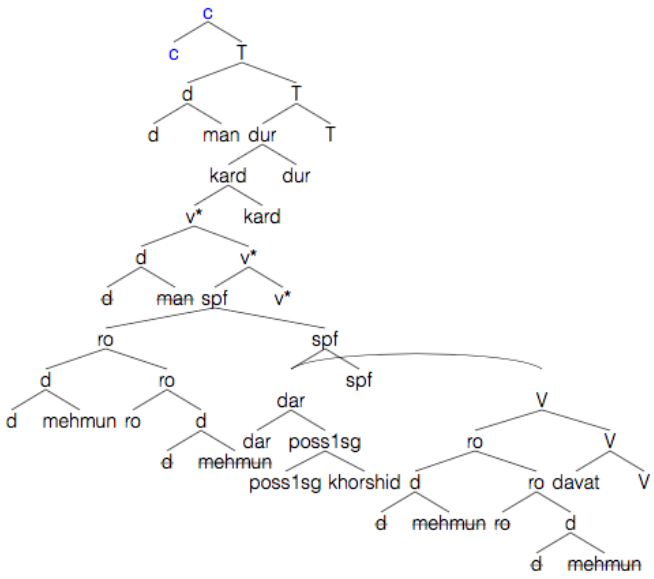
Merge [c] and [T[d[d..][man..]][T[dur..][T..]]]
Label from [c] (syntactic head merging with a non-head)

Stream:	[]
Stack (⇓):	[d[d][man]] [ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]] [d[d][mehmun]]
SO:	[c[c][T[d[d][man]][T[dur[kard[v*[d[d][man]][v*[spf[ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]][spf<[dar[dar][poss1sg[poss1sg][khorshid]]],[V[ro[d[d][m 

Specific Object Movement example

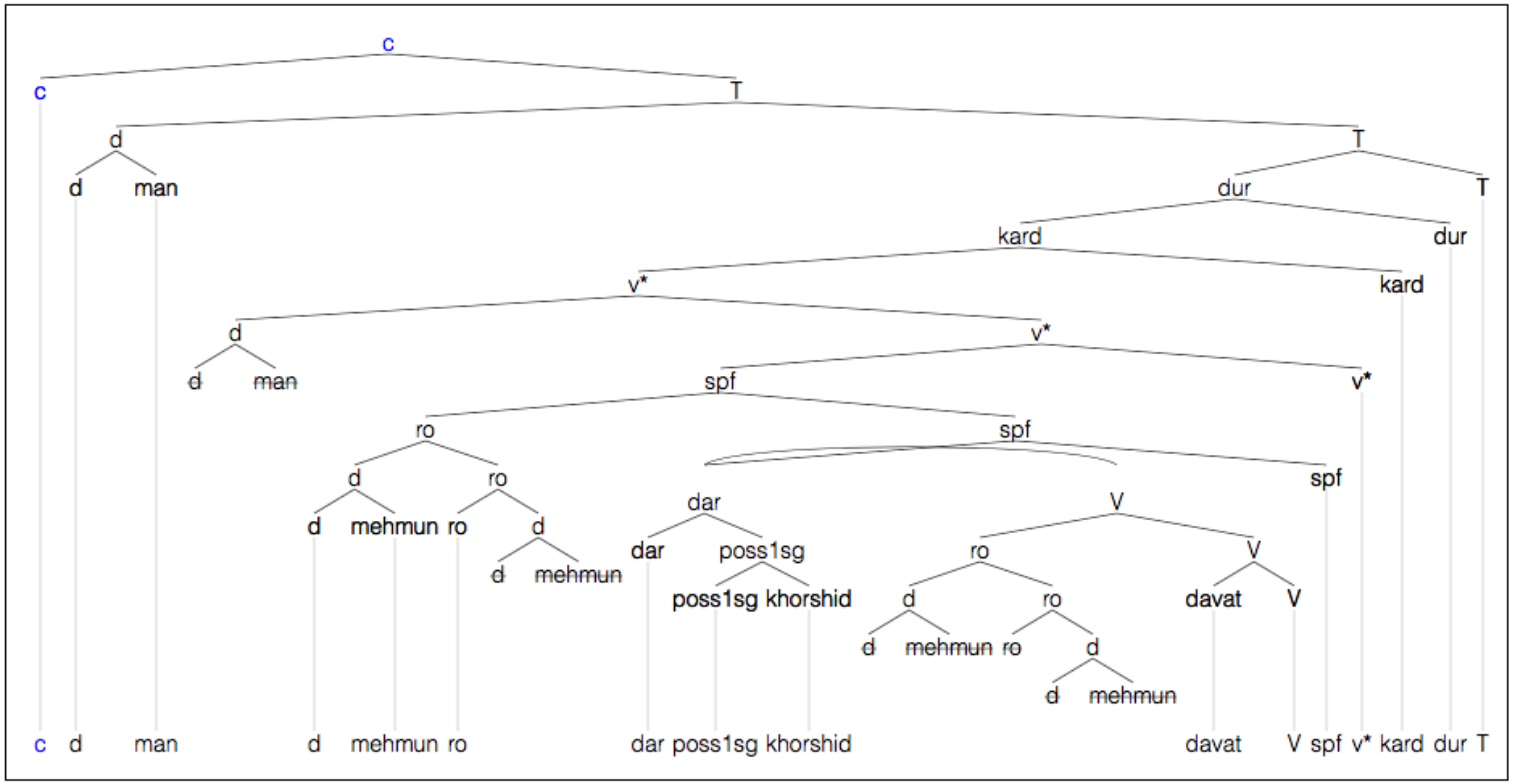
Step 29.

Local Extent boundary at [c[c][T[d..][T..]]]
New boundary (b) marker stacked

Stream:	[]
Stack (⇓):	b [d[d][man]] [ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]] [d[d][mehmun]]
SO:	[c[c][T[d[d][man]][T[dur[kard[v*[d[d][man]][v*[spf[ro[d[d][mehmun]][ro[ro][d[d][mehmun]]]][spf<[dar[dar][poss1sg[poss1sg][khorshid]]],[V[ro[d[d][mehmun]][ro 

Specific Object Movement example

Parse:



Spell-out:
man mehmun ro dar -am khorshid davat kard mi= t(pres,[1,sg]) (after morpheme realization)
man mehmun ro dar khorshid -am davat mi= kard t(pres,[1,sg]) (after affix-hop)
man mehmun ro dar khorshid -am davat mi= kon #am (after morpheme realization, stage 2)
man mehmun ro dar khorshid-am davat mi-kon-am

Unsolved Problems with Minimalism and Persian

Future Tense

- The Future Tense is marked with what seems to be a verbal object "khâstan" that gets the subject's ϕ-features and appears between the NVE and LV, strongly implying a head-movement relation.

Future Tense

- The Future Tense is marked with what seems to be a verbal object "khâstan" that gets the subject's ϕ-features and appears between the NVE and LV, strongly implying a head-movement relation.

(30)

Bahâr	miz-â-ro	tamiz	khâhad-Ø	kard
B.	table-PL-SPF	clean	want.PRS-3.SG	do.PST

'Bahar will clean the tables'

Complement Clause Extraposition

- Despite being strongly head-final in the verbal complex, Persian internal CPs appear at the end of a sentence. Not only at the end, but after *all* verbal items, including stacked auxiliaries (in dialects that allow them).

Complement Clause Extraposition

- Despite being strongly head-final in the verbal complex, Persian internal CPs appear at the end of a sentence. Not only at the end, but after *all* verbal items, including stacked auxiliaries (in dialects that allow them).

(32) Bahâr goft-Ø ke Reza zabun yâd gereft-Ø
B. say.PST-3.SG that R. language memory take.PST-3SG

'Bahar said that Reza learned the language'

Complement Clause Extraposition

- Despite being strongly head-final in the verbal complex, Persian internal CPs appear at the end of a sentence. Not only at the end, but after *all* verbal items, including stacked auxiliaries (in dialects that allow them).

(33) Bahâr goft-Ø ke Reza zabun yâd gereft-Ø
B. say.PST-3.SG that R. language memory take.PST-3SG
'Bahar said that Reza learned the language'

- This problem also exists in German, and to the best of my knowledge, there is no satisfying solution.

Conclusion

Summary

It is possible to build a functional grammar for Persian in a Minimalist Program framework, but it isn't easy.

Additional allowances for Morphology need to be made in above the pure Minimalist Program.

A new rule for the spell-out of affixes is proposed, donating a new testable hypothesis to an otherwise not-very-scientific framework.

And even then, minimalism not quite good enough to describe every aspect of the language (but still an improvement over willy-nilly phrase-structure grammars, and a worthwhile step forward towards finding UG).

Questions?

Backup slides

English Example

English Example

Derivation:

(5) John was arrested

Stream:	[[john!D],[d!case!N],[arrest],[prt!phi!case],[v~],[Tpast!phi],[c]]
Stack (⇓):	
SO:	[]

English Example

Step 1.

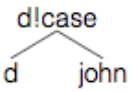
Head of stream [john!D] is the initial SO

Stream:	[[d!case!N],[arrest],[prt!phi!case],[v~],[Tpast!phi],[c]]
Stack (⇓):	
SO:	[john!D] john!D

English Example

Step 2.

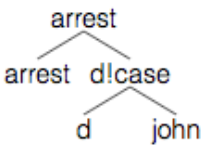
Merge [d!case!N] and [john!D]
Label from [d!case!N] (head with an unvalued uF)
Inherit interpretable feature(s) [f(phi,[3,sg,n])] from [john!D]
[d!case!N] values D on [john!D]
[john] values N on [d!case!N]

Stream:	[[arrest],[prt!phi!case],[v~],[Tpast!phi],[c]]
Stack (⇓):	
SO:	[d!case[d][john]] 

English Example

Step 3.

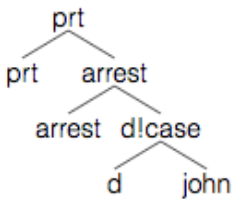
Merge [arrest] and [d!case[d][john]]
Label from [arrest] (head merging with a non-head)
Theta-mark [d!case[d][john]]
Push [d!case[d][john]] (unvalued uF) onto stack

Stream:	[[prt!phi!case],[v~],[Tpast!phi],[c]]
Stack (⇓):	[d!case[d][john]]
SO:	[arrest[arrest][d!case[d][john]]] 

English Example

Step 4.

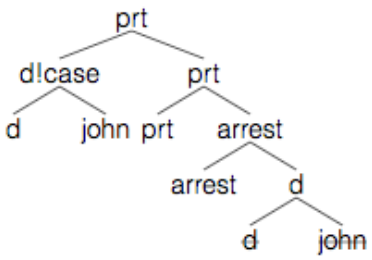
Merge [prt!phi!case] and [arrest[arrest][d!case[d..][john..]]]
Label from [prt!phi!case] (head merging with a non-head)
[d!case[d][john]] values uPhi on [prt!phi!case]
Unified case feature on [prt!case] and [d!case[d][john]]

Stream:	[[v~],[Tpast!phi],[c]]
Stack (⇓):	[d!case[d][john]]
SO:	[prt[prt][arrest[arrest][d!case[d][john]]]] 

English Example

Step 5.

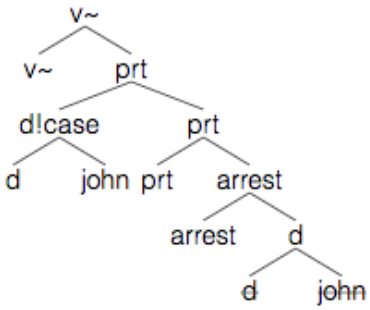
Internal merge selected
Merge [prt[prt][arrest[arrest..][d..]]] and [d!case[d][john]]
Label from [prt[prt][arrest[arrest..][d..]]] (edge feature)

Stream:	[[v~],[Tpast!phi],[c]]
Stack (⇓):	[d!case[d][john]]
SO:	[prt[d!case[d][john]][prt[prt][arrest[arrest][d[d][john]]]]] 

English Example

Step 6.

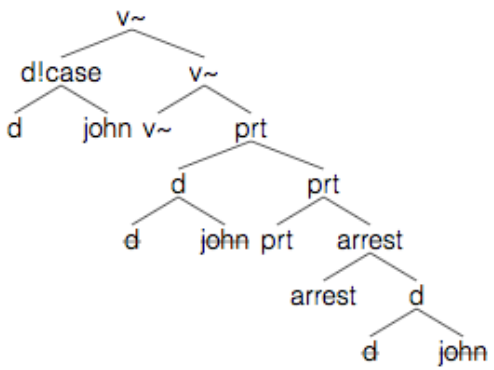
Merge [v~] and [prt[d!case[d..][john..]][prt[prt..][arrest..]]]
Label from [v~] (head merging with a non-head)
[v~] checks theta on [d!case[d][john]]

Stream:	[[Tpast!phi],[c]]
Stack (⇓):	[d!case[d][john]]
SO:	[v~[v~][prt[d!case[d][john]][prt[prt][arrest[arrest][d[d][john]]]]]] 

English Example

Step 7.

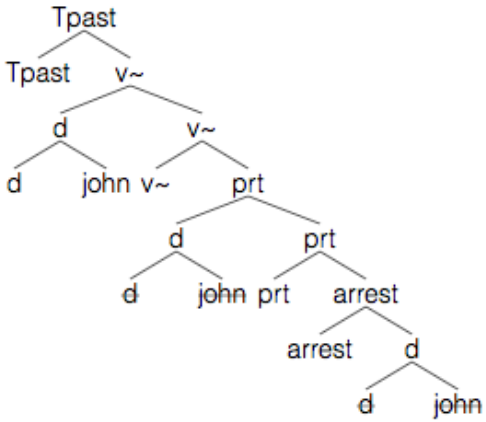
Internal merge selected
Merge [v~[v~][prt[d..][prt..]]] and [d!case[d][john]]
Label from [v~[v~][prt[d..][prt..]]] (edge feature)

Stream:	[[Tpast!phi],[c]]
Stack (⇓):	[d!case[d][john]]
SO:	[v~[d!case[d][john]][v~[v~][prt[d[d][john]][prt[prt][arrest[arrest][d[d][john]]]]]]] 

English Example

Step 8.

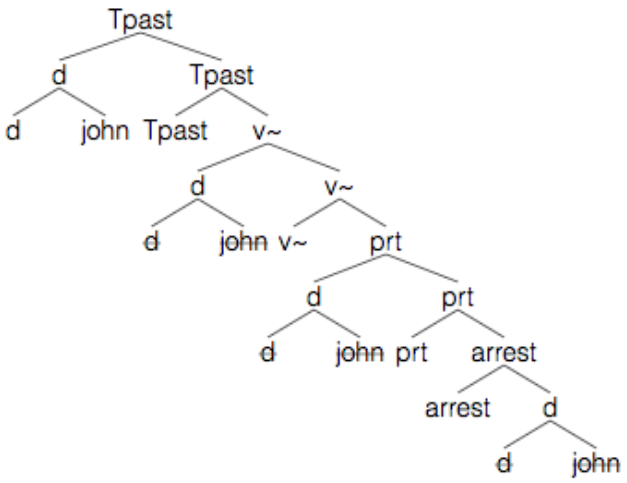
Merge [Tpast!phi] and [v~[d!case[d..][john..]][v~[v~..][prt..]]]
Label from [Tpast!phi] (head merging with a non-head)
[d!case[d][john]] values uPhi on [Tpast!phi]
[Tpast] values nom case on [d!case[d][john]]

Stream:	[[c]]
Stack (⇓):	[d[d][john]]
SO:	[Tpast[Tpast][v~[d[d][john]][v~[v~][prt[d[d][john]][prt[prt][arrest[arrest][d[d][john]]]]]]]] 

English Example

Step 9.

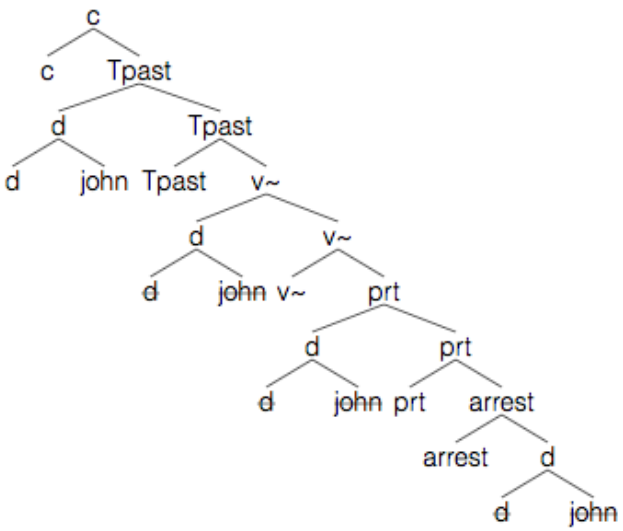
Internal merge selected
Merge [Tpast[Tpast][v~[d..][v~..]]] and [d[d][john]]
Label from [Tpast[Tpast][v~[d..][v~..]]] (edge feature)

Stream:	[[c]]
Stack (⇓):	[d[d][john]]
SO:	[Tpast[d[d][john]][Tpast[Tpast][v~[d[d][john]][v~[v~][prt[d[d][john]][prt[prt][arrest[arrest][d[d][john]]]]]]]] 

English Example

Step 10.

Merge [c] and [Tpast[d[d..][john..]][Tpast[Tpast..][v~..]]]
Label from [c] (head merging with a non-head)

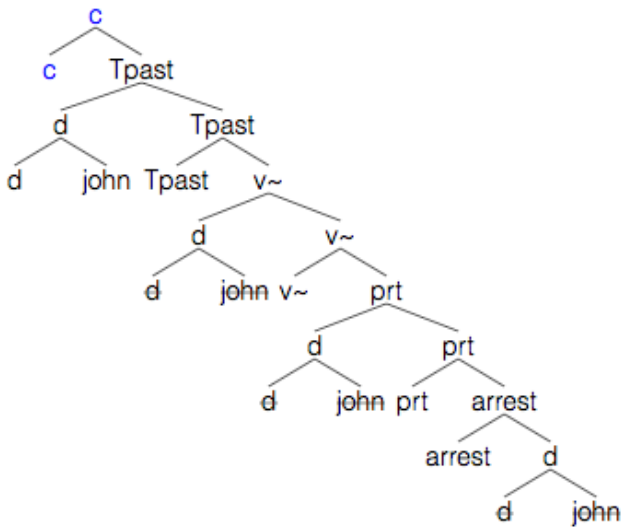
Stream:	[]
Stack (⇓):	[d[d][john]]
SO:	[c[c][Tpast[d[d][john]][Tpast[Tpast][v~[d[d][john]][v~[v~][prt[d[d][john]][prt[prt][arrest[arrest][d[d][john]]]]]]]]] 

English Example

Step 11.

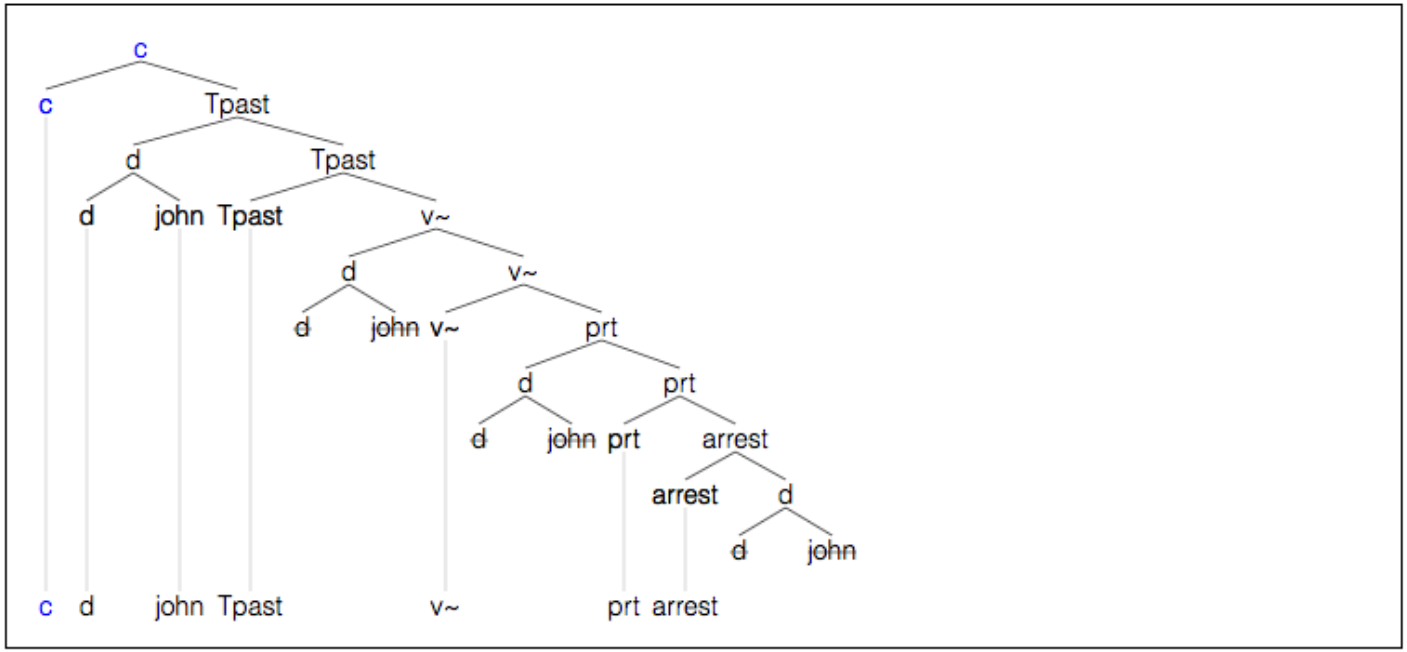
Local Extent boundary at [c[c][Tpast[d..][Tpast..]]]
New boundary (b) marker stacked

Stream:	[]
Stack (⇓):	b [d[d][john]]
SO:	[c[c][Tpast[d[d][john]][Tpast[Tpast][v~[d[d][john]][v~[v~][prt[d[d][john]][prt[prt][arrest[arrest][d[d][john]]]]]]]]]]



English Example

Parse:



Spell-out:
john -ed(sg) be -en arrest (after morpheme realization)
john be -ed(sg) arrest -en (after affix-hop)
john be -ed(sg) arrest -en (after morpheme realization, stage 2)
john was arrested

Syntax 1:01

Syntax 1:01

- Language is not linear in nature

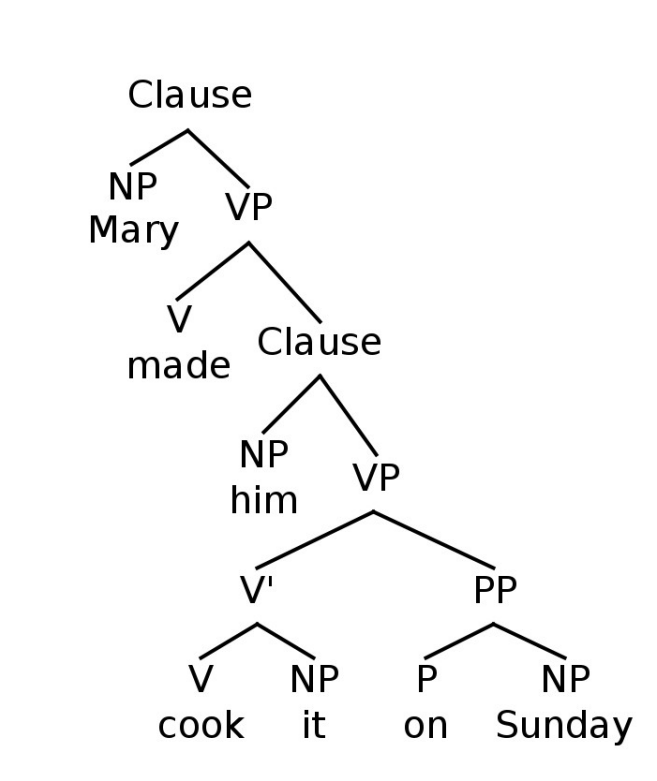
Syntax 1:01

- Language is not linear in nature
  - *Eagles that fly instinctively swim*

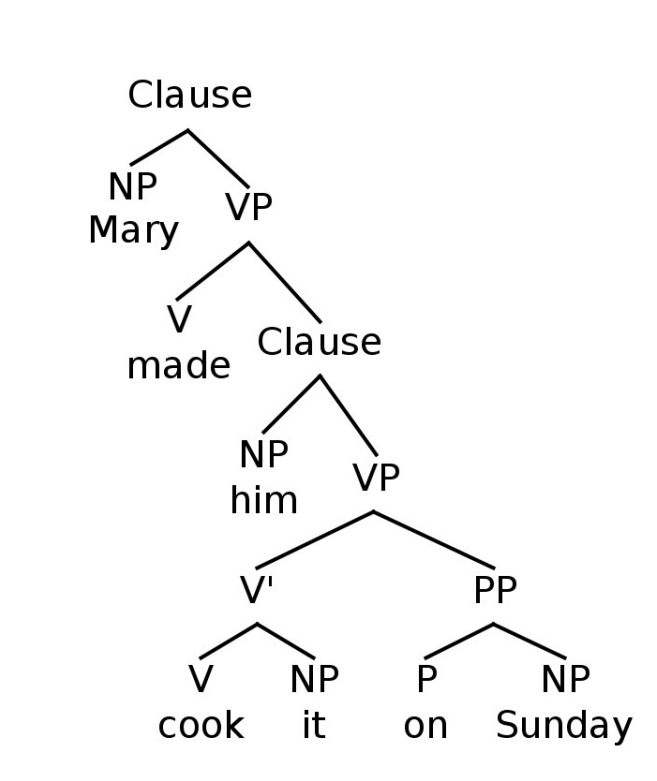
Syntax 1:01

- Language is not linear in nature
  - *Eagles that fly instinctively swim*
  - *Instinctively eagles that fly swim*

Syntax 1:01

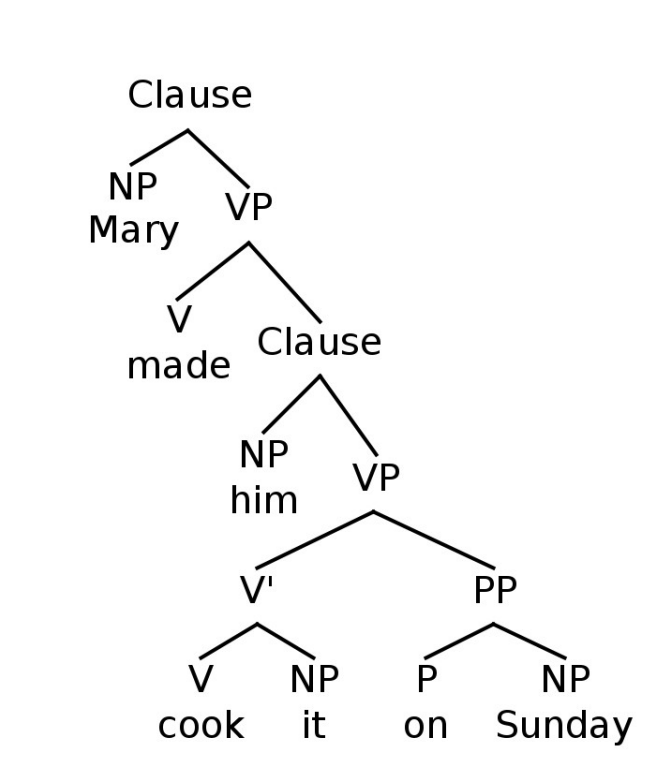

Clause
NP
Mary
VP
V
made
Clause
NP
him
VP
V'
PP
V
cook
NP
it
P
on
NP
Sunday

Syntax 1:01



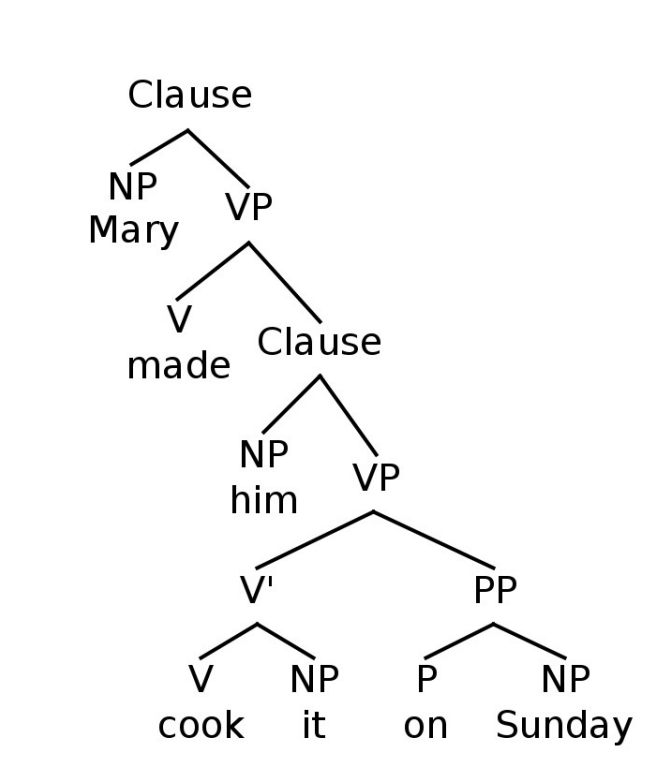
- Binary branching

Syntax 1:01



- Binary branching
- Phrases and heads

Syntax 1:01



- Binary branching
- Phrases and heads
- Movement

Syntax 1:01

Does this work for other languages?

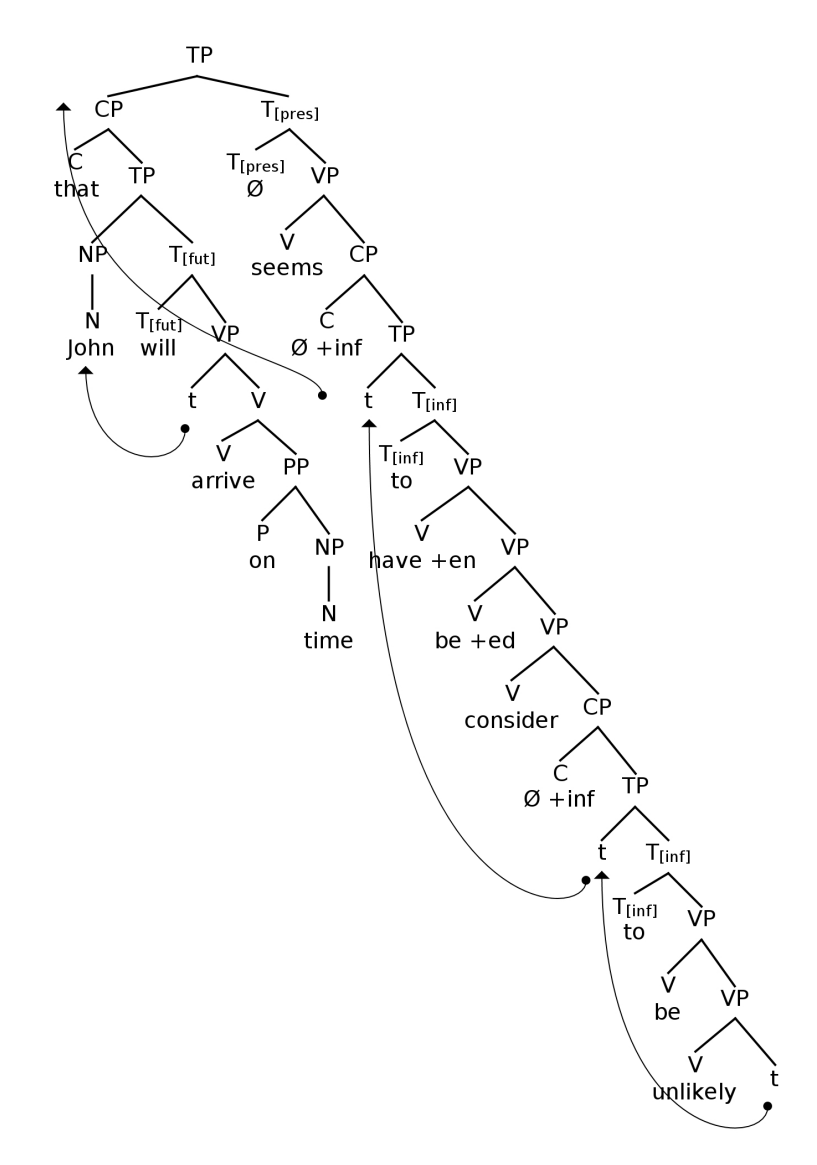
Syntax 1:01

Does this work for other languages?

kinda?

Syntax 1:01

That John will arrive on time seems to have been considered to be unlikely.



Edge Features

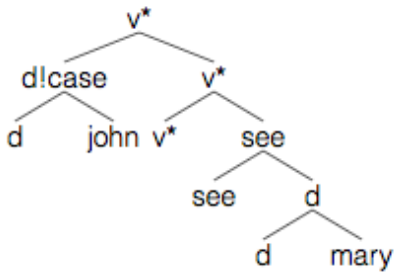
Edge features probe for the first constituent available, or the last one to have uninterpretable features still unvalued

Edge Features

Edge features probe for the first constituent available, or the last one to have uninterpretable features still unvalued

Step 9.

Merge [v*[v*][see[see..][d..]]] and [d!case[d][john]]
Label from [v*[v*][see[see..][d..]]] (edge feature)
Theta-mark [d!case[d][john]]
Push [d!case[d][john]] (unvalued uF) onto stack

Stream:	[[Tpast!phi],[c]]
Stack (⇓):	[d!case[d][john]] [d[d][mary]]
SO:	[v*[d!case[d][john]][v*[v*][see[see][d[d][mary]]]]] 

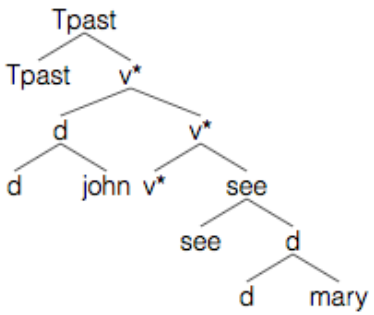
This is realized in the Minimalist Machine with a stack onto which is pushed any item that does not have all its features valued.

Edge Features

Edge features probe for the first constituent available, or the last one to have uninterpretable features still unvalued

Step 10.

Merge [Tpast!phi] and [v*[d!case[d..][john..]][v*[v*..][see..]]]
Label from [Tpast!phi] (head merging with a non-head)
[d!case[d][john]] values uPhi on [Tpast!phi]
[Tpast] values nom case on [d!case[d][john]]

Stream:	[[c]]
Stack (⇓):	[d[d][john]] [d[d][mary]]
SO:	[Tpast[Tpast][v*[d[d][john]][v*[v*][see[see][d[d][mary]]]]]] 

This is realized in the Minimalist Machine with a stack onto which is pushed any item that does not have all its features valued.

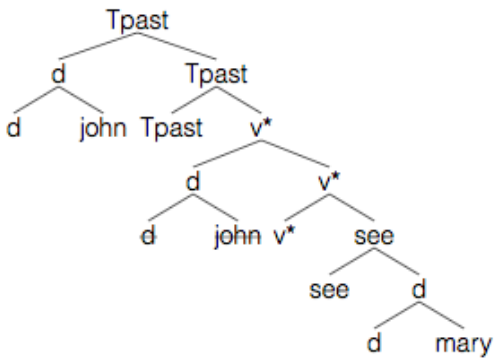
Edge Features

Edge features probe for the first constituent available, or the last one to have uninterpretable features still unvalued

Step 11.

Internal merge selected
Merge [Tpast[Tpast][v*[d..][v*..]]] and [d[d][john]]
Label from [Tpast[Tpast][v*[d..][v*..]]] (edge feature)

Stream:	[[c]]
Stack (⇓):	[d[d][john]] [d[d][mary]]
SO:	[Tpast[d[d][john]][Tpast[Tpast][v*[d[d][john]][v*[v*][see[see][d[d][mary]]]]]]]



This is realized in the Minimalist Machine with a stack onto which is pushed any item that does not have all its features valued.

References I

N. Chomsky.
A minimalist program for linguistic theory.
The view from Building 20: Essays in linguistics in honor of Sylvain Bromberger, 1:1–52, 1993.

N. Chomsky.
Derivation by phase.
Ken Hale: A Life in Language, 1:1–52, 2001.

S. Fong and J. Ginsburg.
Minimalist machine.
In *University of Arizona SynSalon*, April 2015.

S. Karimi.
***A Minimalist Approach to Scrambling*.**
Addison-Wesley, Reading, MA, 2005.

References II

S. Karimi, M. Mahdavi, R. Nabors, R. Smith, and T. Sullivan.
To do ellipsis or to not do: Constraints on ellipsis in iranian complex predicates.
preprint, 2016.