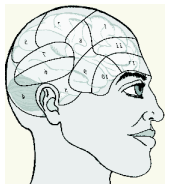
Tractor

Toward Deep Understanding of Short Intelligence Messages

Stuart C. Shapiro

Department of Computer Science and Engineering
Center for Cognitive Science,
Center for Multisource Information Fusion
University at Buffalo, The State University of New York
Buffalo, New York 14260
shapiro@buffalo.edu



Outline

1. Acknowledgements
2. Introduction
3. Syntactic Processing
4. Human Coreference Editing
5. From Annotations to Syntactic Propositional Graph
6. CBIR
7. SNePS 3 and its GUI
8. Syntax-Semantics Mapping
9. Statistics
10. Performance on Motivational Example
11. Summary
12. Further Reading

Acknowledgements

Daniel R. Schlegel, James M. Dobler
Michael Prentice, Paul Bunter, Michael W. Kandefer

Rakesh Nagi, Kedar Sambhoos, Geoff A. Gross, Greg Tauer

Multidisciplinary University Research Initiative (MURI) Grant
(Number W911NF-09-1-0392)
for “Unified Research on Network-based Hard/Soft Information Fusion”
Issued by the US Army Research Office (ARO)
under the program management of Dr. John Lavery.

Tractor I/O

Input: A short English message

- Mostly 1–3 sentences.
- In Counter-insurgency domain.
- Written by human informant or intelligence gatherer.
- Not necessarily “grammatical” English.

Output: Semantic Propositional Graph

- Representing information in the message.
- Nodes for Entities, Events, Actions, Categories, Properties, Property Values, Propositions, ...
- Edges represent non-conceptual relations.

Tractor I/O

Input: A short English message

- Mostly 1–3 sentences.
- In Counter-insurgency domain.
- Written by human informant or intelligence gatherer.
- Not necessarily “grammatical” English.

Output: Semantic Propositional Graph

- Representing information in the message.
- Nodes for Entities, Events, Actions, Categories, Properties, Property Values, Propositions, ...
- Edges represent non-conceptual relations.

Important Assumption

- A single message is written by a single person at a single time.
- Different messages might be written by different people at different times,
  without the author of one message being aware of the contents of previous messages.
- Therefore,
  - NLP techniques are appropriate for intra-message coreference resolution,
  - inter-message coreference resolution must be based on semantic descriptions of the various entities.
- Tractor must supply those semantic descriptions.

Important Assumption

- A single message is written by a single person at a single time.
- Different messages might be written by different people at different times,
  without the author of one message being aware of the contents of previous messages.
- Therefore,
  - NLP techniques are appropriate for intra-message coreference resolution,
  - inter-message coreference resolution must be based on semantic descriptions of the various entities.
- Tractor must supply those semantic descriptions.

Important Assumption

- A single message is written by a single person at a single time.
- Different messages might be written by different people at different times,
  without the author of one message being aware of the contents of previous messages.
- Therefore,
  - NLP techniques are appropriate for intra-message coreference resolution,
  - inter-message coreference resolution must be based on semantic descriptions of the various entities.
- Tractor must supply those semantic descriptions.

Important Assumption

- A single message is written by a single person at a single time.
- Different messages might be written by different people at different times,
  without the author of one message being aware of the contents of previous messages.
- Therefore,
  - NLP techniques are appropriate for intra-message coreference resolution,
  - inter-message coreference resolution must be based on semantic descriptions of the various entities.
- Tractor must supply those semantic descriptions.

Important Assumption

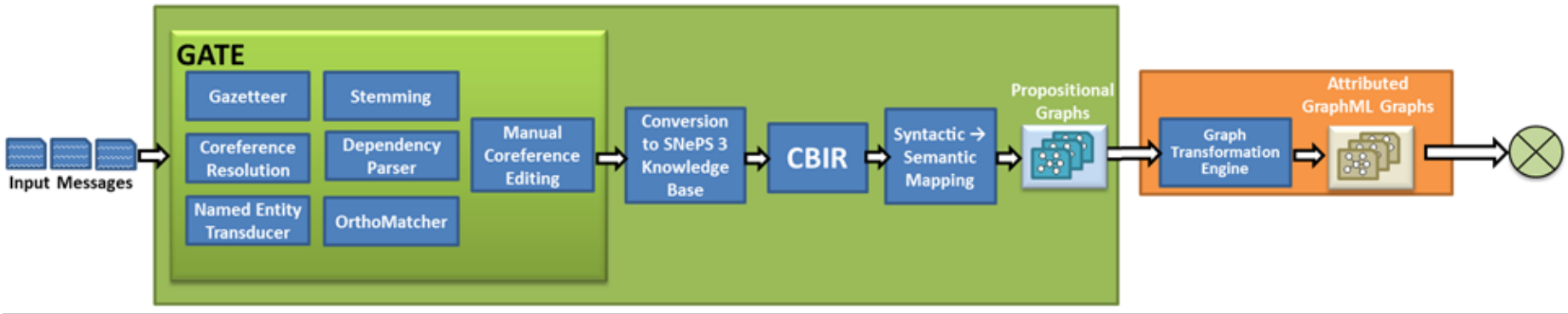
- A single message is written by a single person at a single time.
- Different messages might be written by different people at different times,
  without the author of one message being aware of the contents of previous messages.
- Therefore,
  - NLP techniques are appropriate for intra-message coreference resolution,
  - inter-message coreference resolution must be based on semantic descriptions of the various entities.
- Tractor must supply those semantic descriptions.

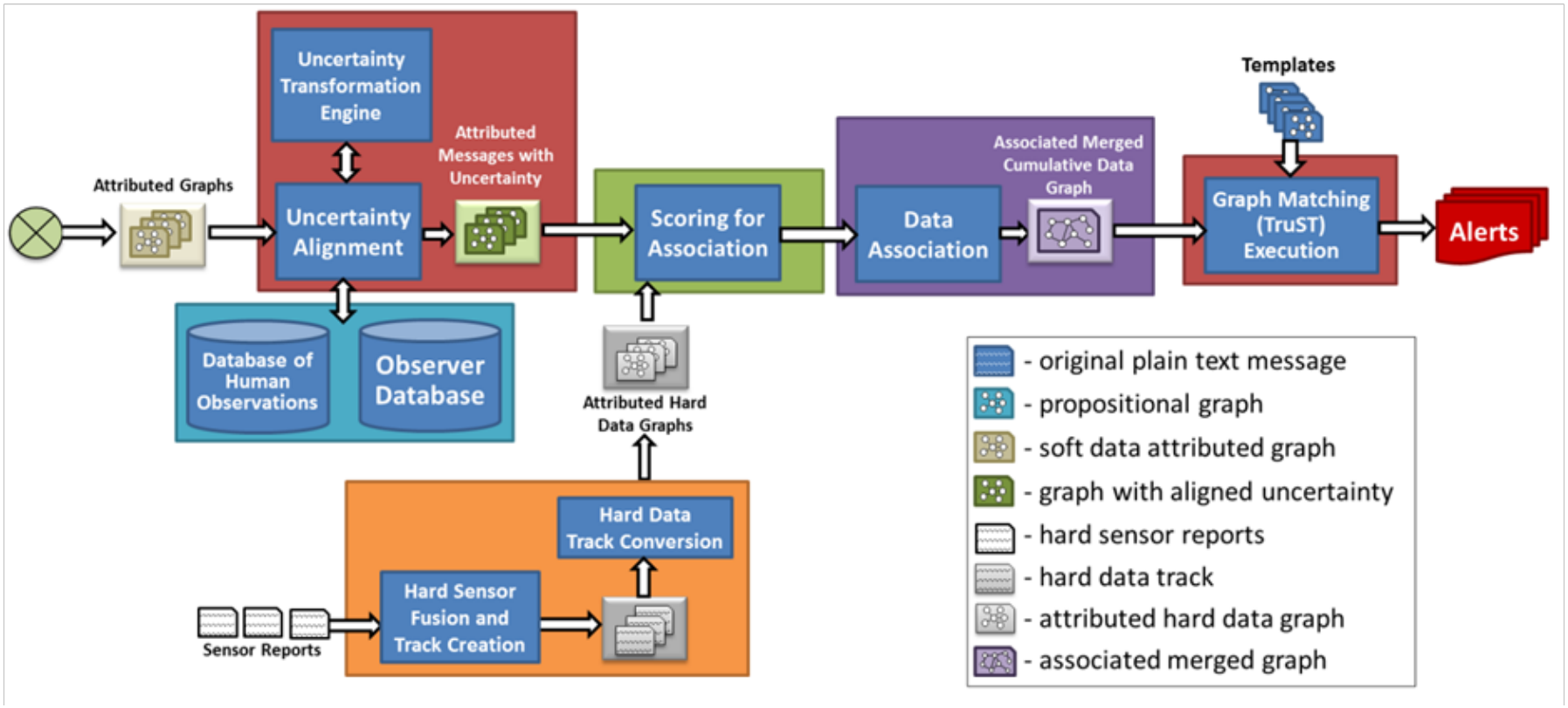
Why Shortness Matters

Since the messages are short,
no need to be concerned with

- rhetorical relations
- topic shifts
- etc.

Hard+Soft Information Fusion Architecture

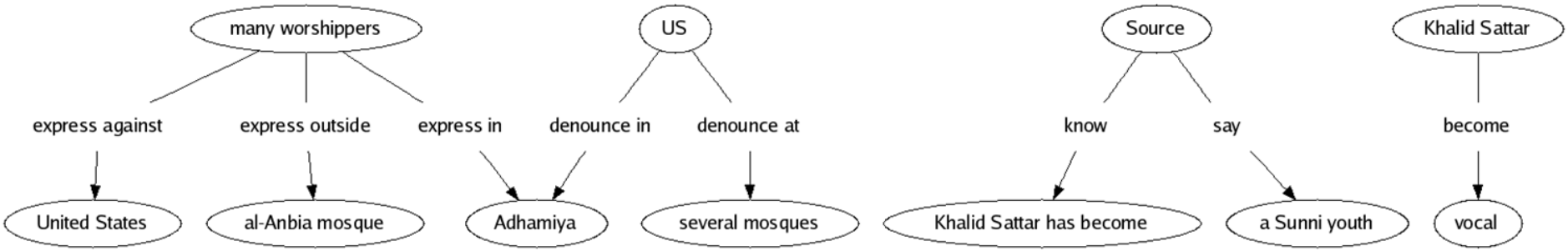




Motivational Example: STEF Messages

- 01/05/07 - Increased hostile sentiment being expressed against U.S. troops by many worshippers outside the al-Anbia mosque in Adhamiya.
- 01/06/07 - Source said a Sunni youth he knows, Khalid Sattar, has become increasingly vocal in denouncing the U.S. at several mosques in Adhamiya.

Motivational Example: STEF Graphs



[K. Sambhoos, J. Llinas, & E. Little, Graphical Methods for Real-Time Fusion and Estimation with Soft Message Data, 11th International Conference on Information Fusion, 2008.]

Questions:

- Relation between “US” and “United States”?
- What was expressed?
- Who denounced whom?
- Does Source know Khalid Sattar?
- Is Khalid Sattar a Sunni youth?
- What did Source say?
- When did these events occur?

Tractor Processing Stream

- Short English Message
- Syntactic Processing
- Human Coreference Editing
- XML Syntactic File of Annotations
- Convert XML to SNePS 3
- SNePS 3 Syntactic Propositional Graph
- Add Contextually Relevant Background/Ontological Information
- Syntax-Semantics Mapping
- SNePS 3 Semantic Propositional Graph
- Express as GraphML
- Semantic Propositional Graph in GraphML Format

Tractor Processing Stream

- Short English Message
- Syntactic Processing
- Human Coreference Editing
- XML Syntactic File of Annotations
- Convert XML to SNePS 3
- SNePS 3 Syntactic Propositional Graph
- Add Contextually Relevant Background/Ontological Information
- Syntax-Semantics Mapping
- SNePS 3 Semantic Propositional Graph
- Express as GraphML
- Semantic Propositional Graph in GraphML Format

Tractor Processing Stream

- Short English Message
- Syntactic Processing
- Human Coreference Editing
- XML Syntactic File of Annotations
- Convert XML to SNePS 3
- SNePS 3 Syntactic Propositional Graph
- Add Contextually Relevant Background/Ontological Information
- Syntax-Semantics Mapping
- SNePS 3 Semantic Propositional Graph
- Express as GraphML
- Semantic Propositional Graph in GraphML Format

Tractor Processing Stream

- Short English Message
- Syntactic Processing
- Human Coreference Editing
- XML Syntactic File of Annotations
- Convert XML to SNePS 3
- SNePS 3 Syntactic Propositional Graph
- Add Contextually Relevant Background/Ontological Information
- Syntax-Semantics Mapping
- SNePS 3 Semantic Propositional Graph
- Express as GraphML
- Semantic Propositional Graph in GraphML Format

Tractor Processing Stream

- Short English Message
- Syntactic Processing
- Human Coreference Editing
- XML Syntactic File of Annotations
- Convert XML to SNePS 3
- SNePS 3 Syntactic Propositional Graph
- Add Contextually Relevant Background/Ontological Information
- Syntax-Semantics Mapping
- SNePS 3 Semantic Propositional Graph
- Express as GraphML
- Semantic Propositional Graph in GraphML Format

Tractor Processing Stream

- Short English Message
- Syntactic Processing
- Human Coreference Editing
- XML Syntactic File of Annotations
- Convert XML to SNePS 3
- SNePS 3 Syntactic Propositional Graph
- Add Contextually Relevant Background/Ontological Information
- Syntax-Semantics Mapping
- SNePS 3 Semantic Propositional Graph
- Express as GraphML
- Semantic Propositional Graph in GraphML Format

Tractor Processing Stream

- Short English Message
- Syntactic Processing
- Human Coreference Editing
- XML Syntactic File of Annotations
- Convert XML to SNePS 3
- SNePS 3 Syntactic Propositional Graph
- Add Contextually Relevant Background/Ontological Information
- Syntax-Semantics Mapping
- SNePS 3 Semantic Propositional Graph
- Express as GraphML
- Semantic Propositional Graph in GraphML Format

Tractor Processing Stream

- Short English Message
- Syntactic Processing
- Human Coreference Editing
- XML Syntactic File of Annotations
- Convert XML to SNePS 3
- SNePS 3 Syntactic Propositional Graph
- Add Contextually Relevant Background/Ontological Information
- Syntax-Semantics Mapping
- SNePS 3 Semantic Propositional Graph
- Express as GraphML
- Semantic Propositional Graph in GraphML Format

Tractor Processing Stream

- Short English Message
- Syntactic Processing
- Human Coreference Editing
- XML Syntactic File of Annotations
- Convert XML to SNePS 3
- SNePS 3 Syntactic Propositional Graph
- Add Contextually Relevant Background/Ontological Information
- Syntax-Semantics Mapping
- SNePS 3 Semantic Propositional Graph
- Express as GraphML
- Semantic Propositional Graph in GraphML Format

Tractor Processing Stream

- Short English Message
- Syntactic Processing
- Human Coreference Editing
- XML Syntactic File of Annotations
- Convert XML to SNePS 3
- SNePS 3 Syntactic Propositional Graph
- Add Contextually Relevant Background/Ontological Information
- Syntax-Semantics Mapping
- SNePS 3 Semantic Propositional Graph
- Express as GraphML
- Semantic Propositional Graph in GraphML Format

Tractor Processing Stream

- Short English Message
- Syntactic Processing
- Human Coreference Editing
- XML Syntactic File of Annotations
- Convert XML to SNePS 3
- SNePS 3 Syntactic Propositional Graph
- Add Contextually Relevant Background/Ontological Information
- Syntax-Semantics Mapping
- SNePS 3 Semantic Propositional Graph
- Express as GraphML
- Semantic Propositional Graph in GraphML Format

Example Message

1. 01/31/2010, 0700 hrs. – Al Sabah newspaper reports that in response to the new government policy, local presidential candidate Azam Al-Azhar has called for a protest at the Second District Courthouse. Al-Azhar said he would personally attend this protest, and that local residents should expect to see his black SUV arrive at the courthouse at around 1800 hrs.

GATE (General Architecture for Text Engineering)

1. Character Sequence to English Text
   1. ANNIE (a Nearly-New Information Extraction System) Tokenizer
   2. ANNIE Sentence Splitter
2. Morphology
   1. ANNIE POS Tagger
   2. English Stemmer
3. Named Entity Recognition
   1. List-Based “Gazetteer”
   2. ANNIE Rule-Base NE Transducer
4. Coreferencers
   1. ANNIE OrthoMatcher
   2. ANNIE Pronominal Coreferencer
5. Stanford Dependency Parser

GATE (General Architecture for Text Engineering)

1. Character Sequence to English Text
   1. ANNIE (a Nearly-New Information Extraction System) Tokenizer
   2. ANNIE Sentence Splitter
2. Morphology
   1. ANNIE POS Tagger
   2. English Stemmer
3. Named Entity Recognition
   1. List-Based “Gazetteer”
   2. ANNIE Rule-Base NE Transducer
4. Coreferencers
   1. ANNIE OrthoMatcher
   2. ANNIE Pronominal Coreferencer
5. Stanford Dependency Parser

GATE (General Architecture for Text Engineering)

1. Character Sequence to English Text
   1. ANNIE (a Nearly-New Information Extraction System) Tokenizer
   2. ANNIE Sentence Splitter
2. Morphology
   1. ANNIE POS Tagger
   2. English Stemmer
3. Named Entity Recognition
   1. List-Based “Gazetteer”
   2. ANNIE Rule-Base NE Transducer
4. Coreferencers
   1. ANNIE OrthoMatcher
   2. ANNIE Pronominal Coreferencer
5. Stanford Dependency Parser

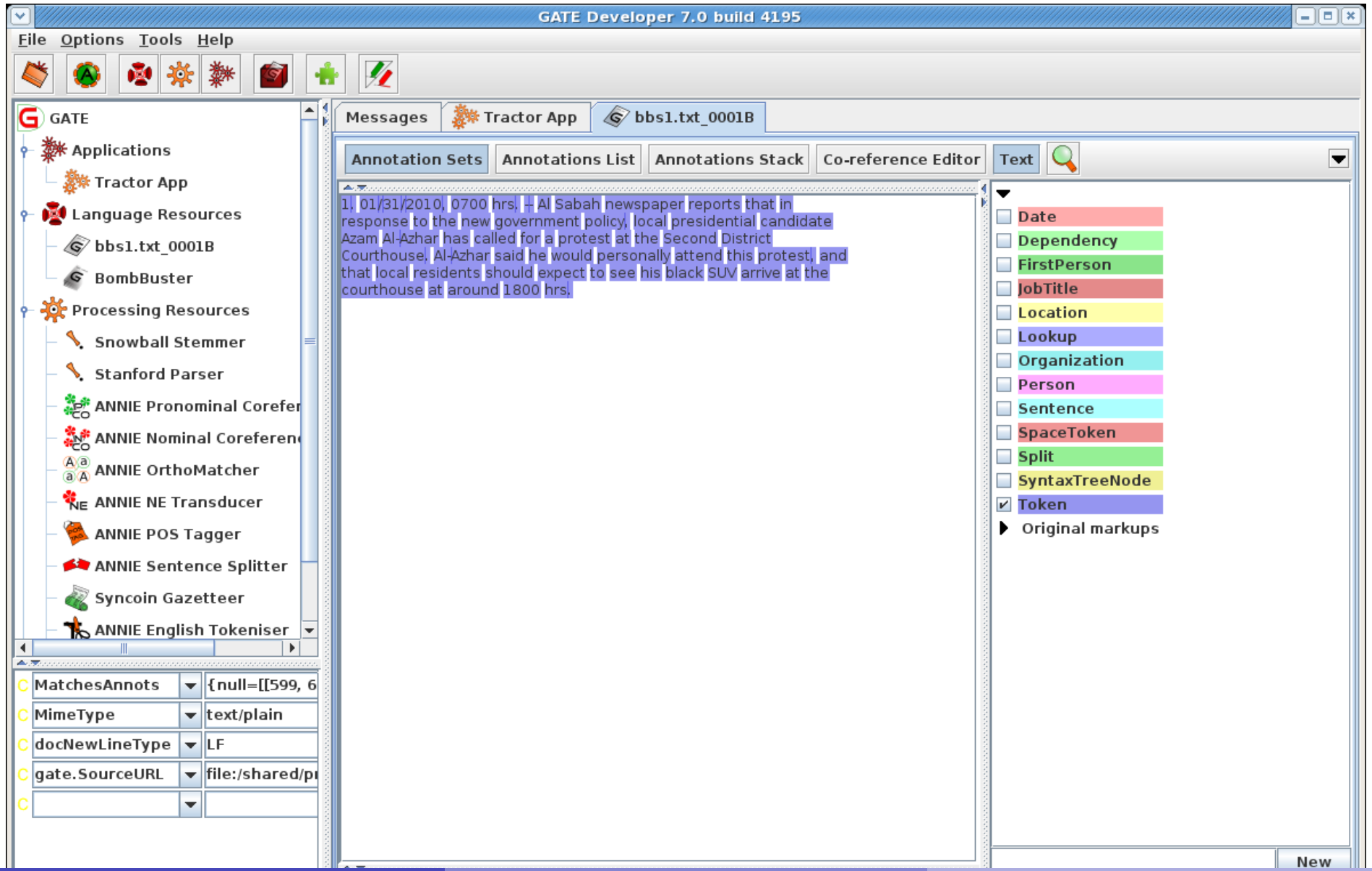
GATE (General Architecture for Text Engineering)

1. Character Sequence to English Text
   1. ANNIE (a Nearly-New Information Extraction System) Tokenizer
   2. ANNIE Sentence Splitter
2. Morphology
   1. ANNIE POS Tagger
   2. English Stemmer
3. Named Entity Recognition
   1. List-Based “Gazetteer”
   2. ANNIE Rule-Base NE Transducer
4. Coreferencers
   1. ANNIE OrthoMatcher
   2. ANNIE Pronominal Coreferencer
5. Stanford Dependency Parser

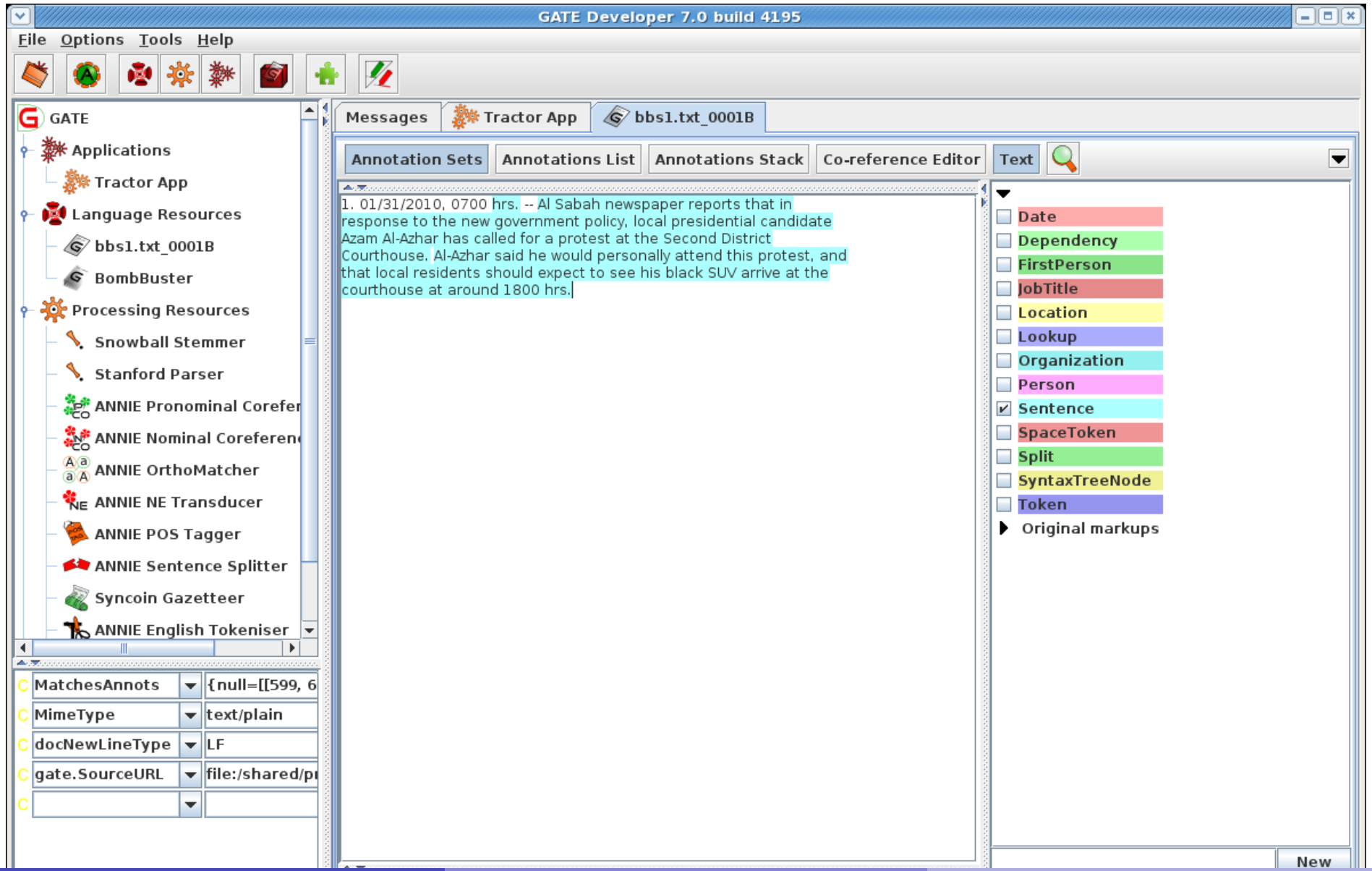
GATE (General Architecture for Text Engineering)

1. Character Sequence to English Text
   1. ANNIE (a Nearly-New Information Extraction System) Tokenizer
   2. ANNIE Sentence Splitter
2. Morphology
   1. ANNIE POS Tagger
   2. English Stemmer
3. Named Entity Recognition
   1. List-Based “Gazetteer”
   2. ANNIE Rule-Base NE Transducer
4. Coreferencers
   1. ANNIE OrthoMatcher
   2. ANNIE Pronominal Coreferencer
5. Stanford Dependency Parser

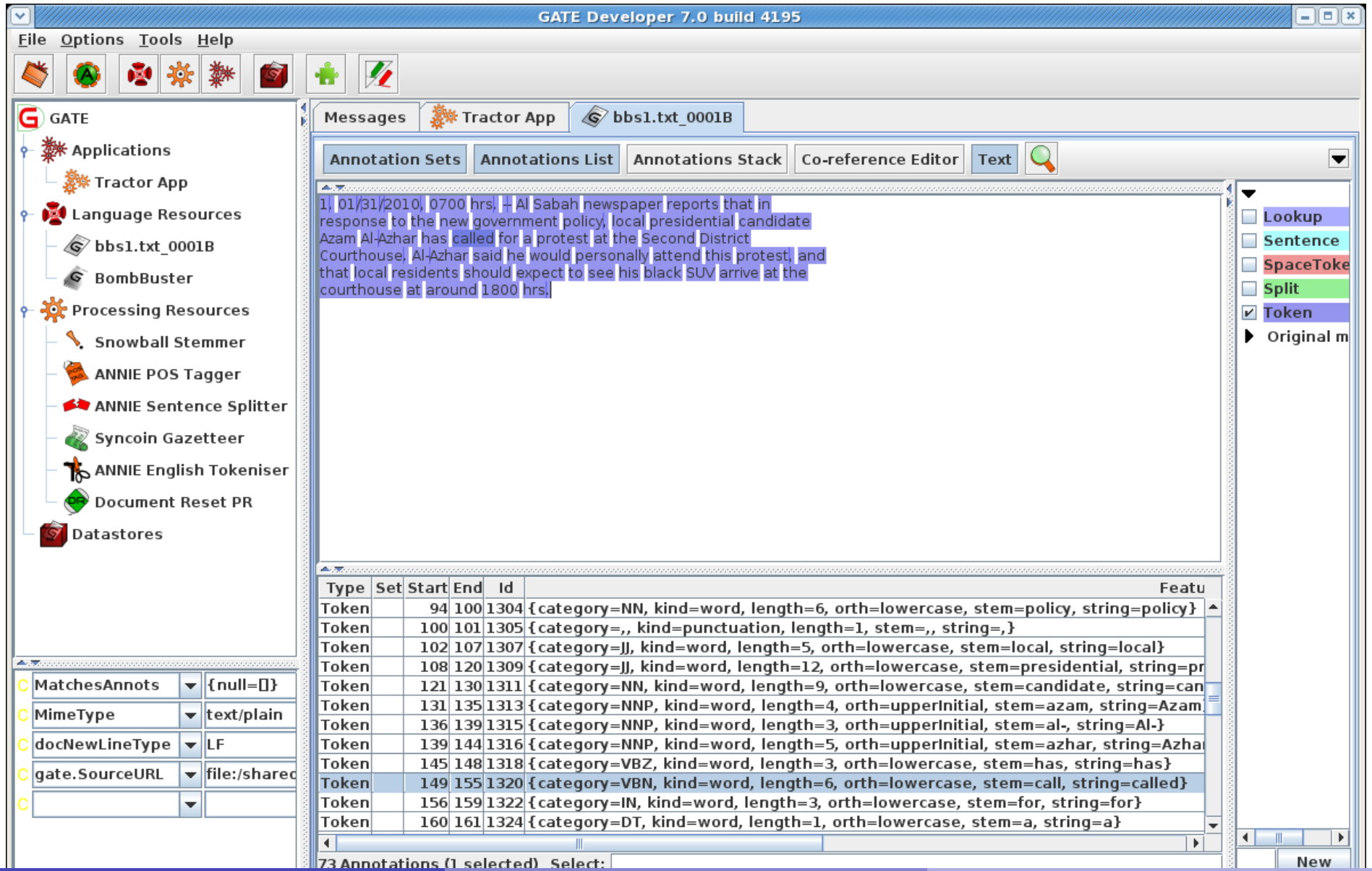
Results of Tokenizer



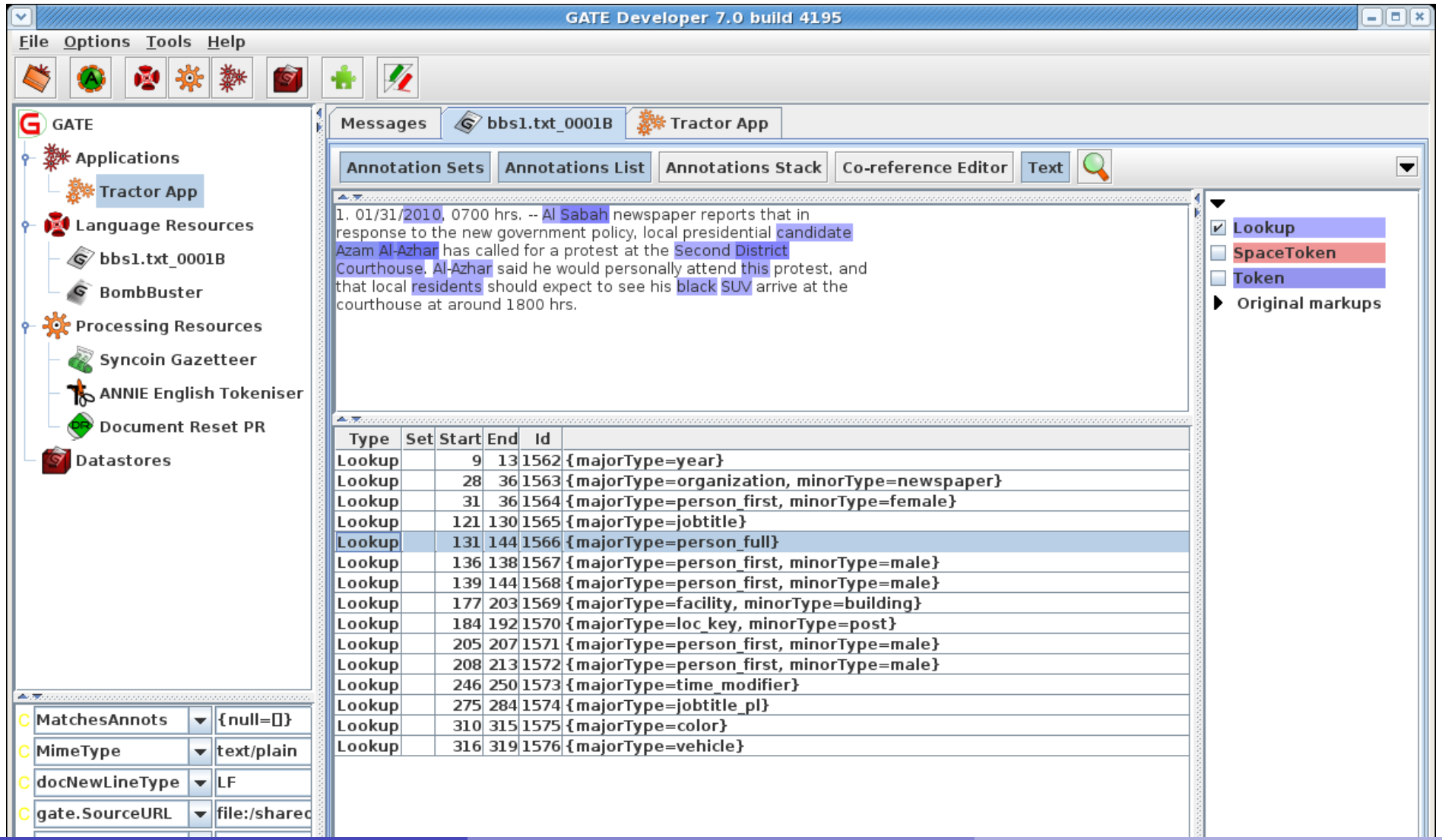
Results of Sentence Splitter



Results of POS Tagger & Stemmer

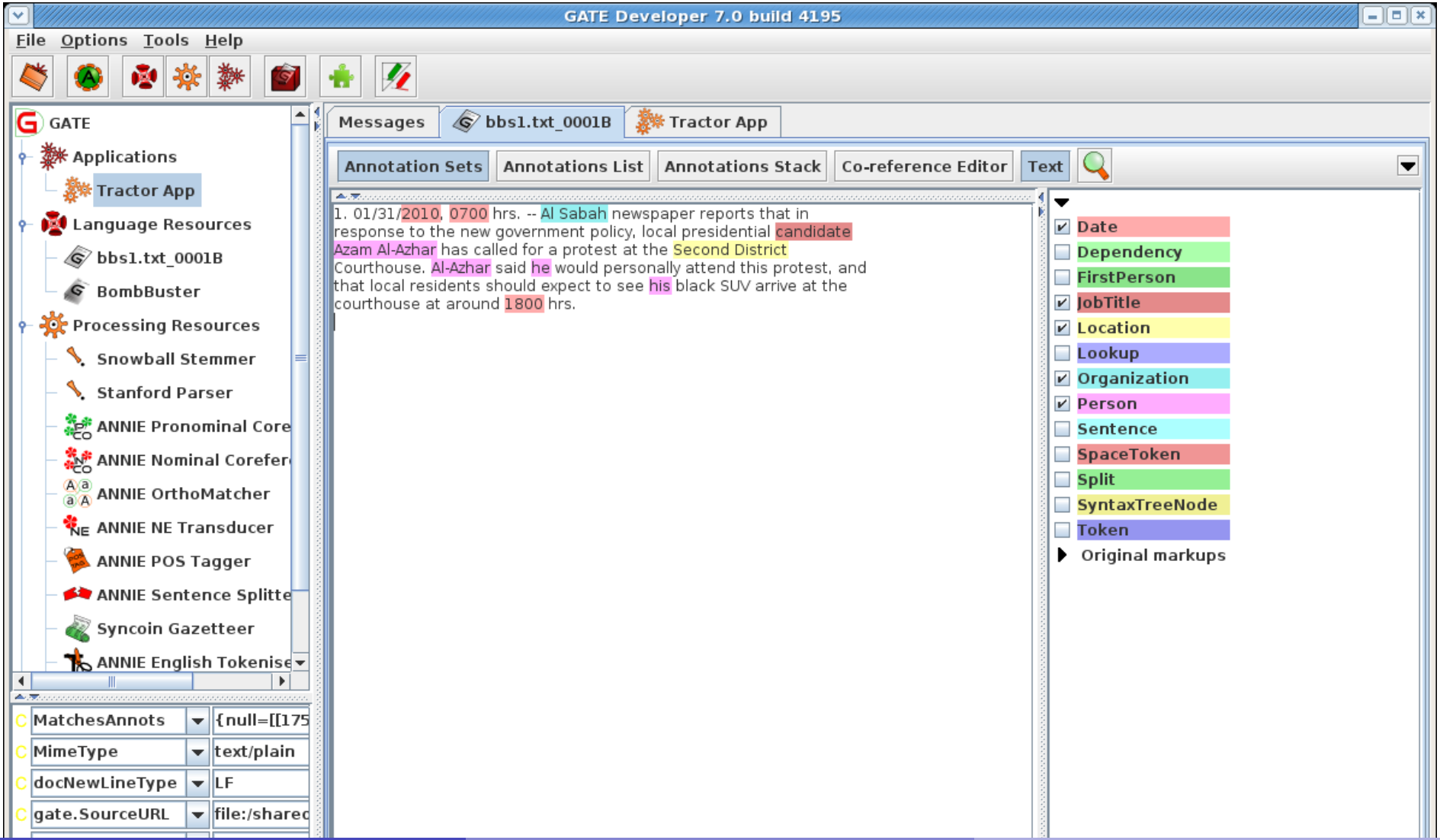


Results of "Gazetteer-Based" Named Entity Recognition

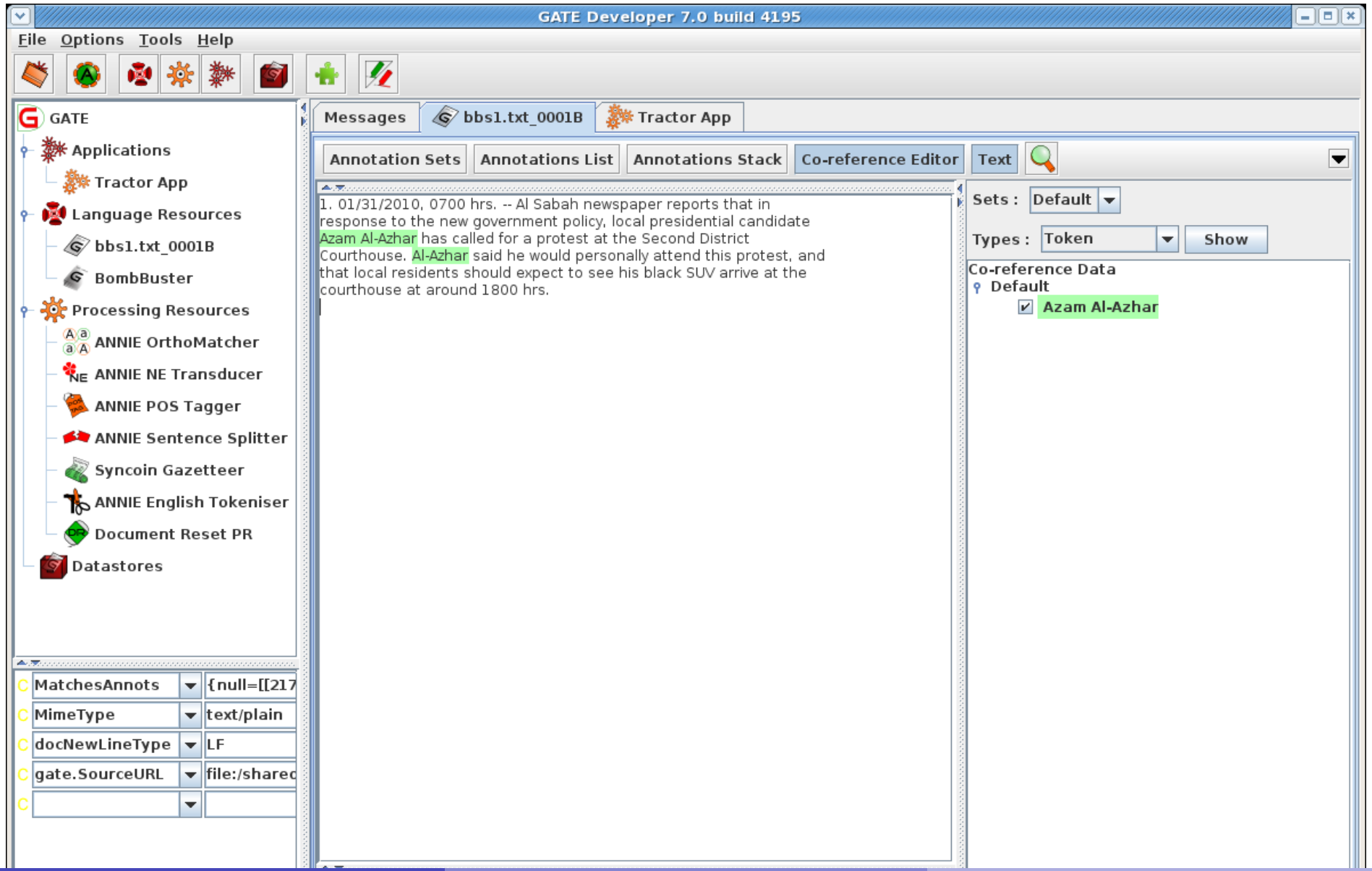


Type	Set	Start	End	Id	
Lookup		9	13	1562	{majorType=year}
Lookup		28	36	1563	{majorType=organization, minorType=newspaper}
Lookup		31	36	1564	{majorType=person_first, minorType=female}
Lookup		121	130	1565	{majorType=jobtitle}
Lookup		131	144	1566	{majorType=person_full}
Lookup		136	138	1567	{majorType=person_first, minorType=male}
Lookup		139	144	1568	{majorType=person_first, minorType=male}
Lookup		177	203	1569	{majorType=facility, minorType=building}
Lookup		184	192	1570	{majorType=loc_key, minorType=post}
Lookup		205	207	1571	{majorType=person_first, minorType=male}
Lookup		208	213	1572	{majorType=person_first, minorType=male}
Lookup		246	250	1573	{majorType=time_modifier}
Lookup		275	284	1574	{majorType=jobtitle_pl}
Lookup		310	315	1575	{majorType=color}
Lookup		316	319	1576	{majorType=vehicle}

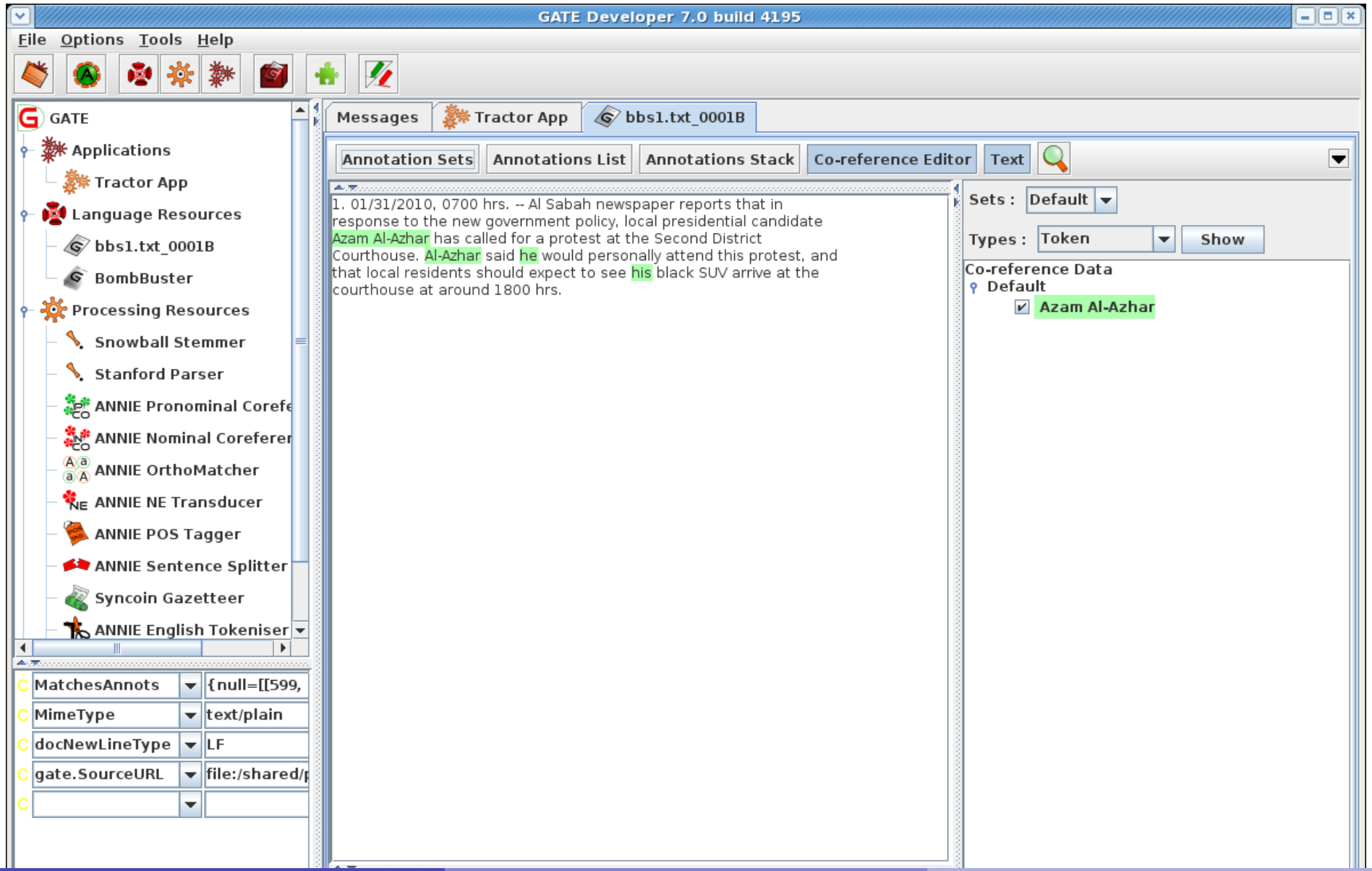
Results of Rule-Based Named Entity Recognition



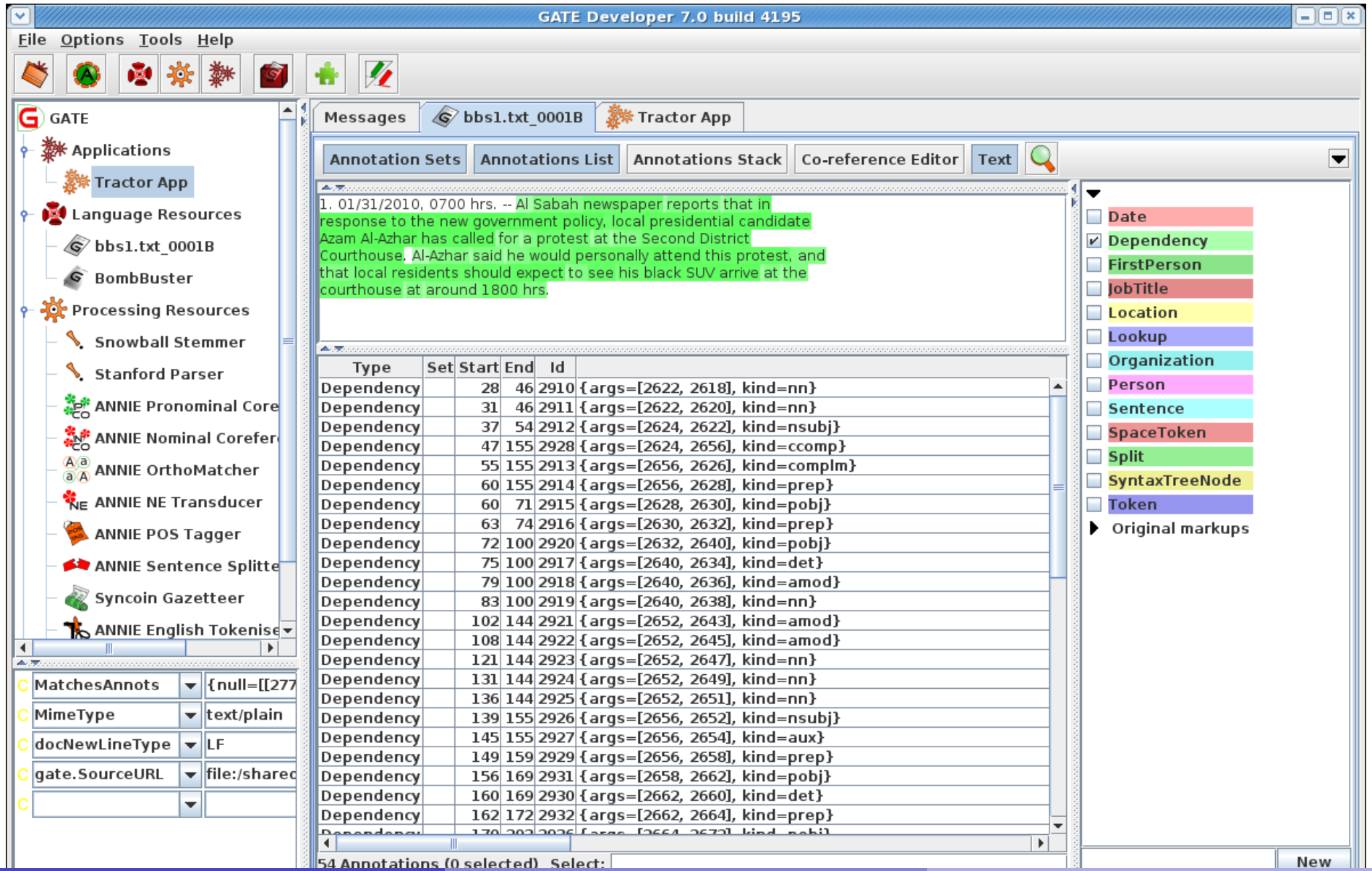
Results of OrthoMatcher



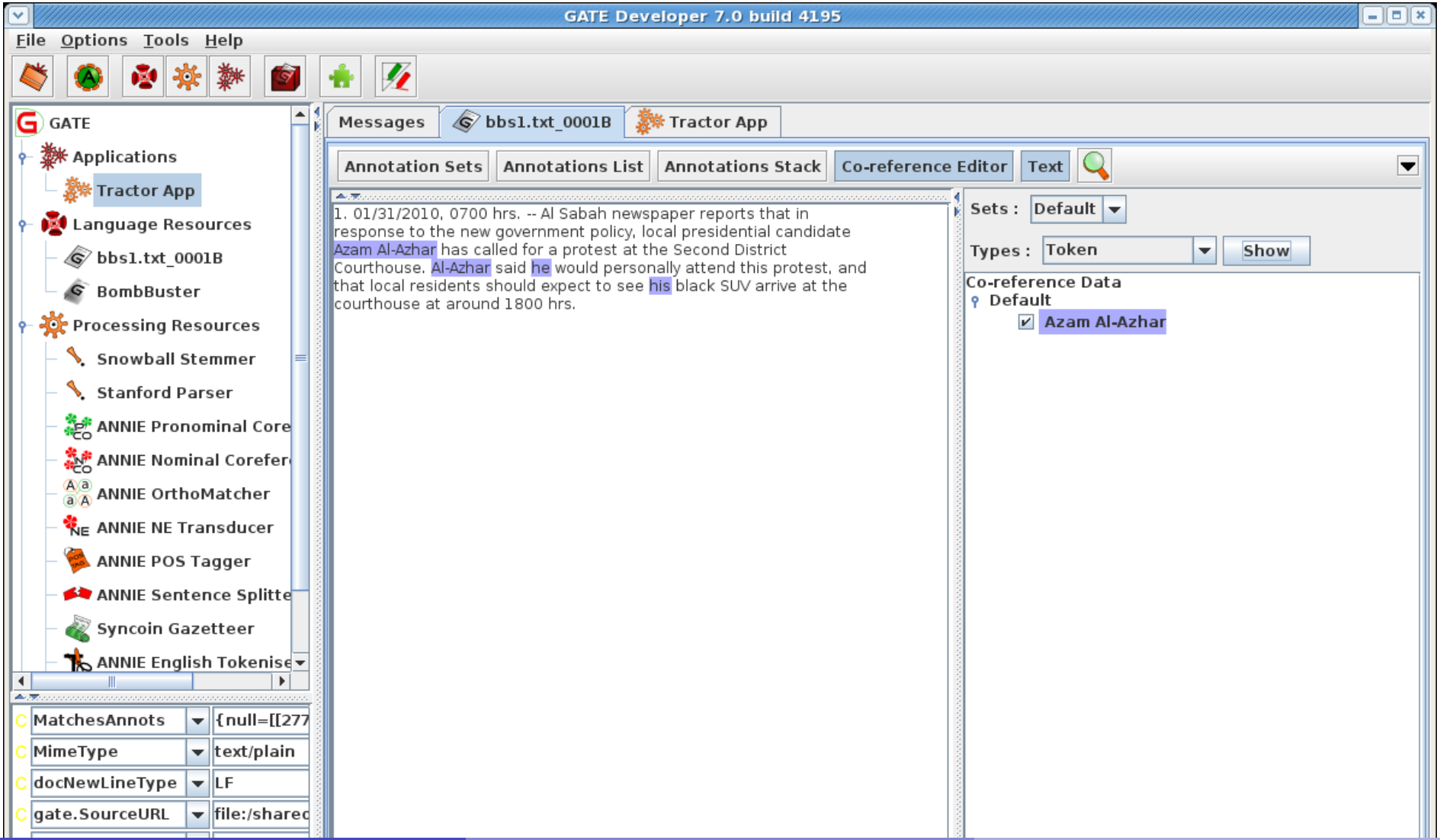
After Pronominal Coreferencer



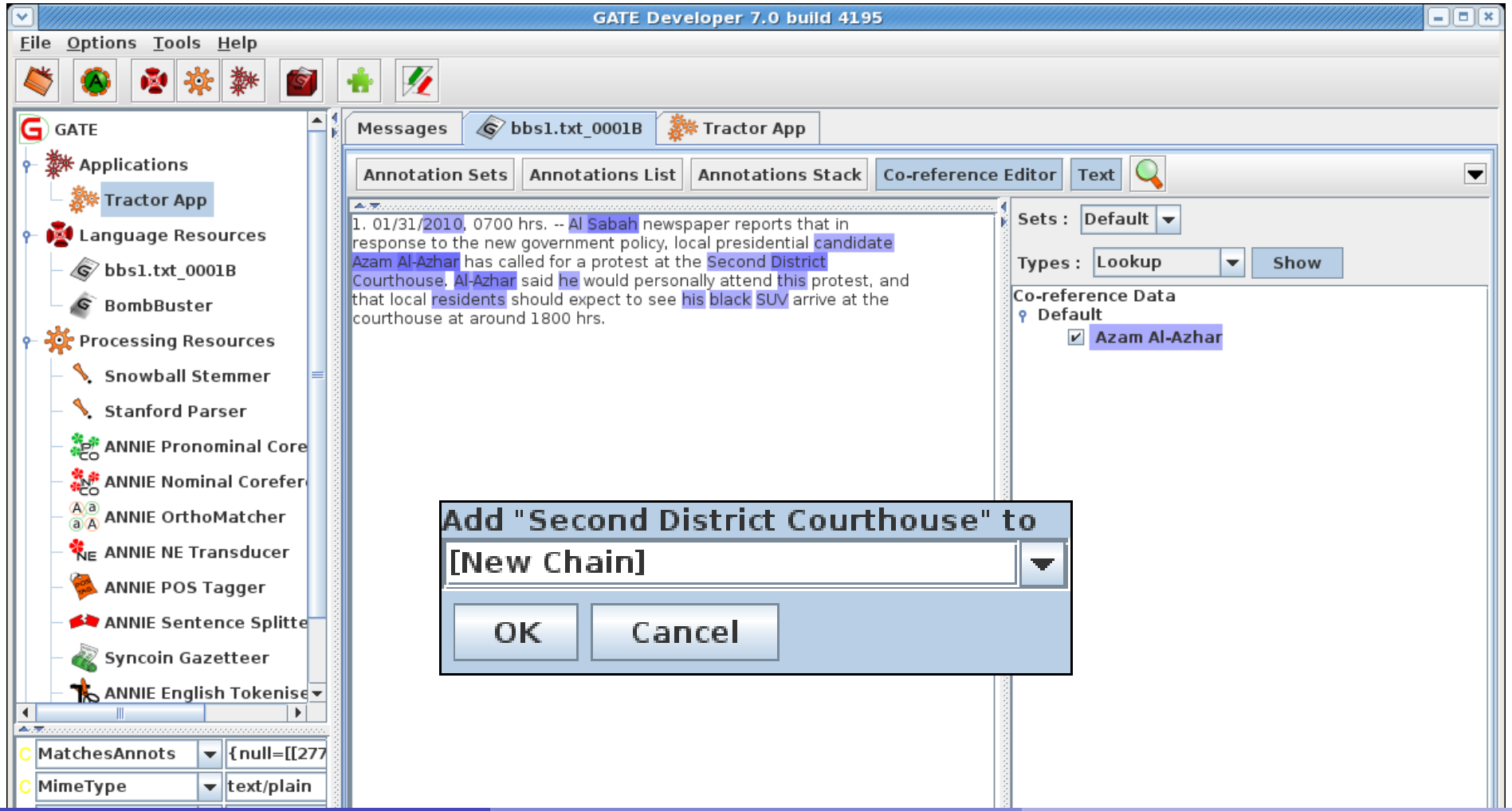
Result of Dependency Parse



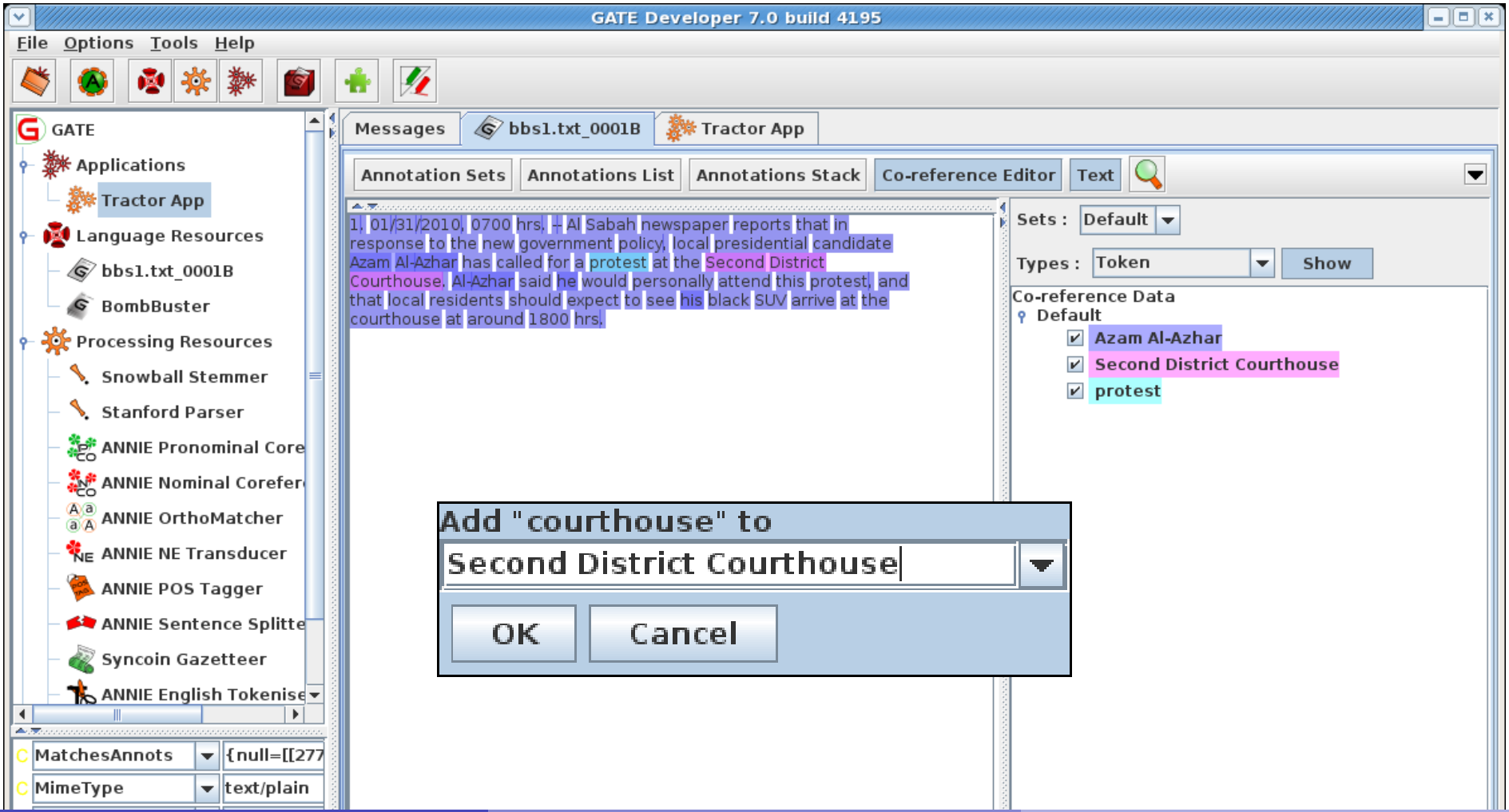
Co-reference Editor
Initial Chain



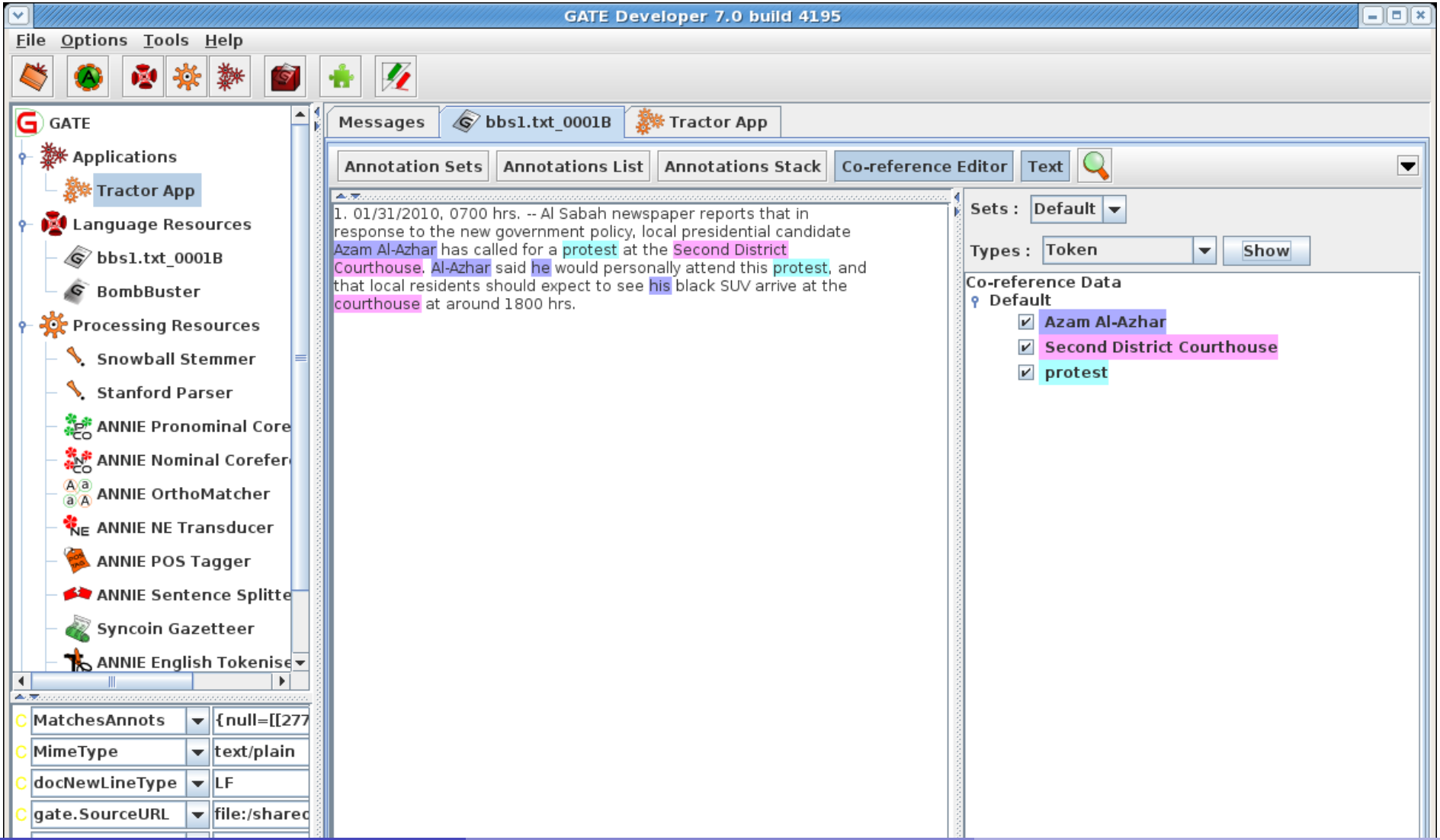
Co-reference Editor
Adding a Chain



Co-reference Editor
Adding to a Chain



Co-reference Editor
Final Chains



XML to SNePS 3

XML

```
<Annotation Id="510" Type="Token" StartNode="251" EndNode="258">
<Feature>
  <Name className="java.lang.String">string</Name>
  <Value className="java.lang.String">protest</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">category</Name>
  <Value className="java.lang.String">NN</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">dependencies</Name>
  <Value className="java.util.ArrayList" itemClassName="java.lang.String">
         det(569)</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">matches</Name>
  <Value className="java.util.ArrayList" itemClassName="java.lang.Integer">
         482;510</Value>
</Feature>
</Annotation>
```

SNePS 3

```
(assert `(token-start-pos 510 251))
(assert `(token-end-pos 510 258))
(assert `(TextOf |protest| 510))
(assert `(SyntacticCategoryOf "NN" 510))
(assert `(det 510 569))
(assert `(Equiv (setof 482 510)))
```

xml2sneps3

Besides translation, `xml2sneps3`

- combines annotations that cover same substring into one token,
- adjusts message times to GMT,
- adjusts message dates as necessary given time change,
- converts message dates to ISO format.

Adding Contextually Relevant Background/Ontological Information

For each noun in the graph

- Finds the Cyc concept from ResearchCyc
- Loads the ontology above it in OpenCyc into the graph

SNePS 3

SNePS 3 is the latest member of the SNePS Family of KRR systems.

It is still being implemented.

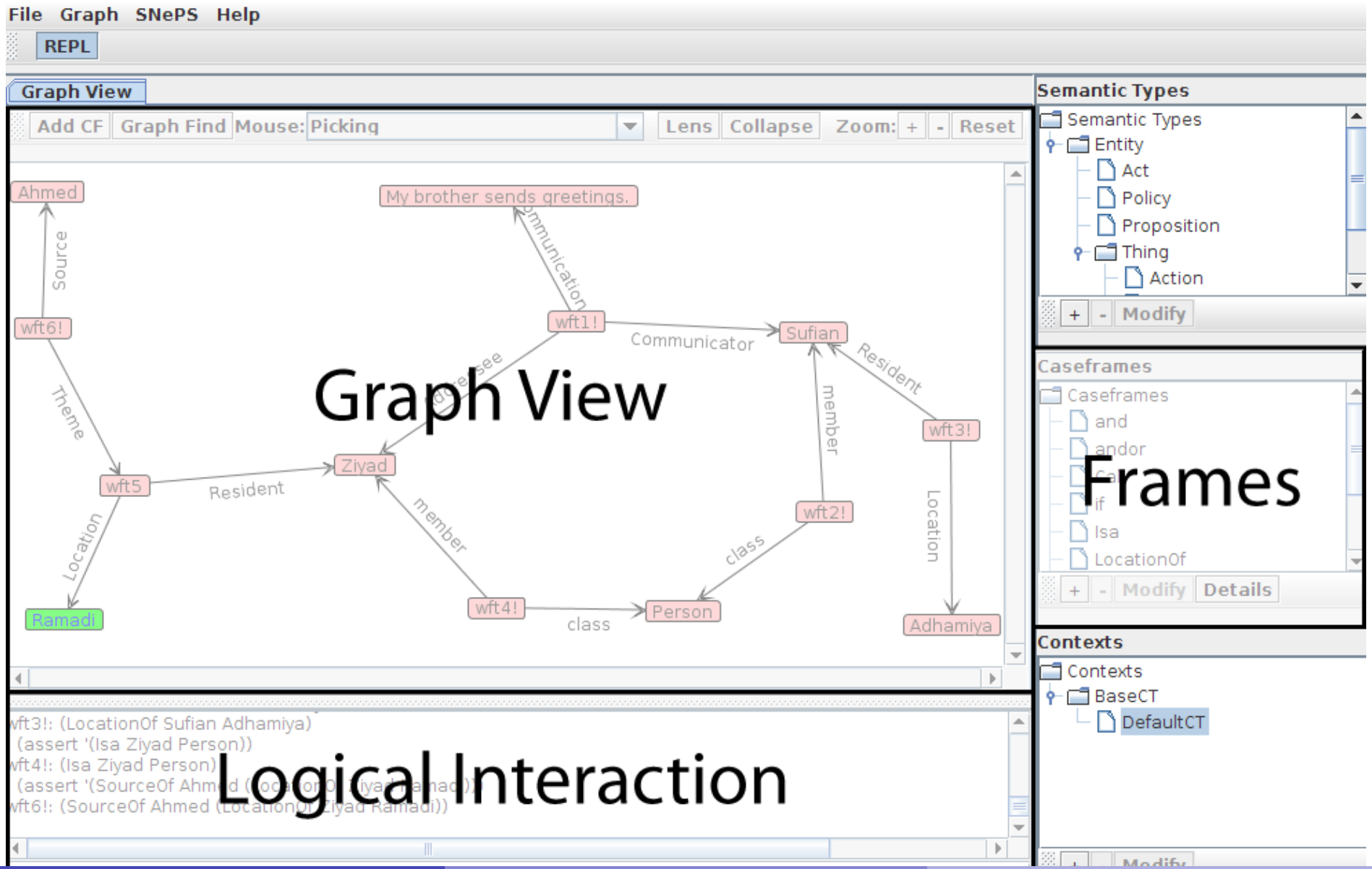
The SNePS 3 KB can be thought of as simultaneously being:

- Logic based,
- Frame based, and
- Graph based.

We have created a user interface which uses all three:

- Assertions and queries of a KB
  are handled using logic or frames.
- Visualization and inspection
  is done using propositional graphs.

SNePS 3 GUI


File Graph SNePS Help
REPL
Graph View
Add CF Graph Find Mouse: Picking Lens Collapse Zoom: + - Reset
Ahmed
My brother sends greetings.
Source
Communication
wft6!
wft1!
Sufian
Communicator
Theme
Resident
wft3!
member
Ziyad
wft5
Resident
wft2!
Location
member
Location
class
wft4!
Person
class
Ramadi
Adhamiya
Graph View
Semantic Types
Semantic Types
Entity
Act
Policy
Proposition
Thing
Action
+ - Modify
Caseframes
Caseframes
and
andor
if
Isa
LocationOf
+ - Modify Details
Frames
Contexts
Contexts
BaseCT
DefaultCT
wft3!: (LocationOf Sufian Adhamiya)
(assert '(Isa Ziyad Person))
wft4!: (Isa Ziyad Person)
wft6!: (SourceOf Ahmed
Logical Interaction

KB as set of Logical Expressions

The SNePS 3 KB is a set of logical expressions:

- Atomic terms
  - Individual constants denoting entities in domain including some relations
- Arbitrary and indefinite terms [Shapiro, KR2004]
- Functional terms including
  - terms denoting atomic propositions
  - terms denoting non-atomic propositions

Use CLIF syntax.
Every logical expression is a term.
Allows propositions about propositions without leaving First-Order.
Internal name of functional terms: `wft`*i* `[!]`
for “well-formed term”.

KB as set of Logical Expressions

The SNePS 3 KB is a set of logical expressions:

- Atomic terms
  - Individual constants denoting entities in domain including some relations
- Arbitrary and indefinite terms [Shapiro, KR2004]
- Functional terms including
  - terms denoting atomic propositions
  - terms denoting non-atomic propositions

Use CLIF syntax.
Every logical expression is a term.
Allows propositions about propositions without leaving First-Order.
Internal name of functional terms: `wft`*i* `[!]`
for “well-formed term”.

KB as set of Logical Expressions

The SNePS 3 KB is a set of logical expressions:

- Atomic terms
  - Individual constants denoting entities in domain including some relations
- Arbitrary and indefinite terms [Shapiro, KR2004]
- Functional terms
  including
  - terms denoting atomic propositions
  - terms denoting non-atomic propositions

Use CLIF syntax.
Every logical expression is a term.
Allows propositions about propositions without leaving First-Order.
Internal name of functional terms: `wft`*i* `[!]`
for “well-formed term”.

KB as set of Logical Expressions

The SNePS 3 KB is a set of logical expressions:

- Atomic terms
  - Individual constants denoting entities in domain including some relations
- Arbitrary and indefinite terms [Shapiro, KR2004]
- Functional terms
  including
  - terms denoting atomic propositions
  - terms denoting non-atomic propositions

Use CLIF syntax.

Every logical expression is a term.

Allows propositions about propositions without leaving First-Order.

Internal name of functional terms: `wft`*i* `[!]`
for “well-formed term”.

KB as set of Logical Expressions

The SNePS 3 KB is a set of logical expressions:

- Atomic terms
  - Individual constants denoting entities in domain including some relations
- Arbitrary and indefinite terms [Shapiro, KR2004]
- Functional terms
  including
  - terms denoting atomic propositions
  - terms denoting non-atomic propositions

Use CLIF syntax.
Every logical expression is a term.
Allows propositions about propositions without leaving First-Order.
Internal name of functional terms: `wft`*i* `[!]`
for “well-formed term”.

KB as set of Logical Expressions

The SNePS 3 KB is a set of logical expressions:

- Atomic terms
  - Individual constants denoting entities in domain including some relations
- Arbitrary and indefinite terms [Shapiro, KR2004]
- Functional terms
  including
  - terms denoting atomic propositions
  - terms denoting non-atomic propositions

Use CLIF syntax.
Every logical expression is a term.
Allows propositions about propositions without leaving First-Order.
Internal name of functional terms: `wft`*i* `[!]`
for “well-formed term”.

Caseframes

- Based on “The Case for Case” [Fillmore, 1968] and The Berkeley FrameNet Project

  [Baker, Fillmore, & Lowe, 1998; Ruppenhofer *et al.*, 2010]

- Frame
  - schematic representation of a situation with a set of participants and conceptual roles.
- Eliminates syntactic differences.
- E.g.
  - Sufian called Ziyad.
  - Ziyad was called by Sufian.
  - a call from Sufian to Ziyad
- We will use “caseframe” for their “frame”
- and use “frame” for an instantiated caseframe.

Caseframes

- Based on “The Case for Case” [Fillmore, 1968] and The Berkeley FrameNet Project

  [Baker, Fillmore, & Lowe, 1998; Ruppenhofer *et al.*, 2010]

- Frame
  - schematic representation of a situation with a set of participants and conceptual roles.
- Eliminates syntactic differences.
- E.g.
  - Sufian called Ziyad.
  - Ziyad was called by Sufian.
  - a call from Sufian to Ziyad
- We will use “caseframe” for their “frame”
- and use “frame” for an instantiated caseframe.

Caseframes

- Based on “The Case for Case” [Fillmore, 1968] and The Berkeley FrameNet Project

  [Baker, Fillmore, & Lowe, 1998; Ruppenhofer *et al.*, 2010]
- Frame
  - schematic representation of a situation with a set of participants and conceptual roles.
- Eliminates syntactic differences.
- E.g.
  - Sufian called Ziyad.
  - Ziyad was called by Sufian.
  - a call from Sufian to Ziyad
- We will use “caseframe” for their “frame”
- and use “frame” for an instantiated caseframe.

Caseframes

- Based on “The Case for Case” [Fillmore, 1968]
  and The Berkeley FrameNet Project

  [Baker, Fillmore, & Lowe, 1998; Ruppenhofer *et al.*, 2010]
- Frame
  - schematic representation of a situation
    with a set of participants
    and conceptual roles.
- Eliminates syntactic differences.
- E.g.
  - Sufian called Ziyad.
  - Ziyad was called by Sufian.
  - a call from Sufian to Ziyad
- We will use “caseframe” for their “frame”
- and use “frame” for an instantiated caseframe.

Components of Caseframes

Definition

A caseframe has

- A name
- A sort
- An ordered list of slots
- ...

Slots

Slots are defined globally
independently of the caseframes that use them.

Definition

A slot has

- A name
- Minimum and maximum number of fillers
- The sort of its fillers
- ...

Examples of Caseframes

Example

`Isa` is a caseframe of type `Proposition`
with slots `member` and `class`.

Example

`Call` is a caseframe of type `Proposition`
with slots `Communicator`, `Addressee`, and `Communication`.

Frames vs. Logical Terms

- A *frame* is an instance of a caseframe.
- The logical term $(\mathtt{F}\ \ x_1, ..., x_n)$
  is represented by an instance of the caseframe named $\mathtt{F}$
  whose slots, $s_1, ..., s_n$
  are filled by the representations of $x_1, ..., x_n$, respectively.

Frames vs. Logical Terms: Example

```
(assert '(Call Sufian Ziyad
                "My brother sends greetings"))
```

creates an instance of the `Call` caseframe

whose `Communicator` slot contains the filler `Sufian`,

whose `Addressee` slot contains `Ziyad`,

and whose `Communication` slot

contains `"My brother sends greetings"`.

Propositional Graphs

A way of visualizing and traversing the frames.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Functional terms (frames)
  - Proposition-denoting functional terms
- Node ID is
  - symbol
  - frame name (`wft`*i* [!])
- Edges drawn
  from the node corresponding to the frame,
  to the nodes corresponding to the slot fillers
- Edges labeled by slot names

Propositional Graphs

A way of visualizing and traversing the frames.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Functional terms (frames)
  - Proposition-denoting functional terms
- Node ID is
  - symbol
  - frame name (`wft`*i* [!])
- Edges drawn
  from the node corresponding to the frame,
  to the nodes corresponding to the slot fillers
- Edges labeled by slot names

Propositional Graphs

A way of visualizing and traversing the frames.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Functional terms (frames)
  - Proposition-denoting functional terms
- Node ID is
  - symbol
  - frame name (`wft`*i* [!])
- Edges drawn
  from the node corresponding to the frame,
  to the nodes corresponding to the slot fillers
- Edges labeled by slot names

Propositional Graphs

A way of visualizing and traversing the frames.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Functional terms (frames)
  - Proposition-denoting functional terms
- Node ID is
  - symbol
  - frame name (`wft`*i* [!])
- Edges drawn
  from the node corresponding to the frame,
  to the nodes corresponding to the slot fillers
- Edges labeled by slot names

Propositional Graphs

A way of visualizing and traversing the frames.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Functional terms (frames)
  - Proposition-denoting functional terms
- Node ID is
  - symbol
  - frame name (`wft`*i* [!])
- Edges drawn
  from the node corresponding to the frame,
  to the nodes corresponding to the slot fillers
- Edges labeled by slot names

Propositional Graphs

A way of visualizing and traversing the frames.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Functional terms (frames)
  - Proposition-denoting functional terms
- Node ID is
  - symbol
  - frame name (`wft`*i* [!])
- Edges drawn
  from the node corresponding to the frame,
  to the nodes corresponding to the slot fillers
- Edges labeled by slot names

Propositional Graphs

A way of visualizing and traversing the frames.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Functional terms (frames)
  - Proposition-denoting functional terms
- Node ID is
  - symbol
  - frame name (`wft`*i* [!])
- Edges drawn
  from the node corresponding to the frame,
  to the nodes corresponding to the slot fillers
- Edges labeled by slot names

Propositional Graphs

A way of visualizing and traversing the frames.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Functional terms (frames)
  - Proposition-denoting functional terms
- Node ID is
  - symbol
  - frame name (`wft`*i* [!])
- Edges drawn
  from the node corresponding to the frame,
  to the nodes corresponding to the slot fillers
- Edges labeled by slot names

Propositional Graphs

A way of visualizing and traversing the frames.

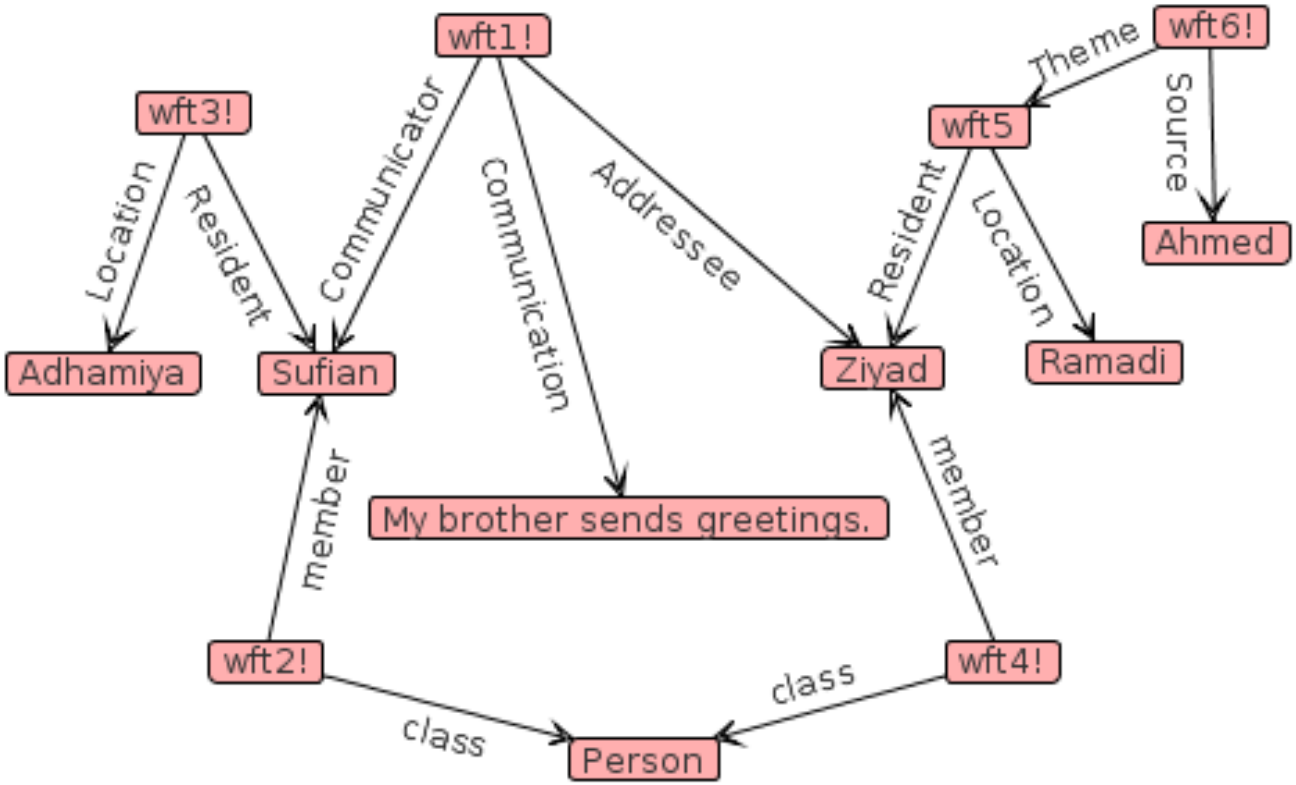
- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Functional terms (frames)
  - Proposition-denoting functional terms
- Node ID is
  - symbol
  - frame name (`wft`*i* [!])
- Edges drawn
  from the node corresponding to the frame,
  to the nodes corresponding to the slot fillers
- Edges labeled by slot names

Propositional Graphs

A way of visualizing and traversing the frames.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Functional terms (frames)
  - Proposition-denoting functional terms
- Node ID is
  - symbol
  - frame name (`wft`*i* [!])
- Edges drawn
  from the node corresponding to the frame,
  to the nodes corresponding to the slot fillers
- Edges labeled by slot names

Example Propositional Graph



"Sufian, a person in Adhamiya, called Ziyad, a person who, according to Ahmed, is in Ramadi, saying 'My brother sends greetings.'"

Graph View as Visualization

- Visualized graph is for human comprehension.
- Visualized graph need not be isomorphic to implementation of KB.
- Usefulness of `wft` nodes:
  - Functional term with more than two arguments (slots).
  - Functional term with more than one filler in a slot.
  - Functional term shown as argument of another (filler in a slot).
- Can show a binary relation with no arc coming into it as a labeled arc ("collapsed arc").

Graph View as Visualization

- Visualized graph is for human comprehension.
- Visualized graph need not be isomorphic to implementation of KB.
- Usefulness of `wft` nodes:
  - Functional term with more than two arguments (slots).
  - Functional term with more than one filler in a slot.
  - Functional term shown as argument of another (filler in a slot).
- Can show a binary relation with no arc coming into it as a labeled arc ("collapsed arc").

Graph View as Visualization

- Visualized graph is for human comprehension.
- Visualized graph need not be isomorphic to implementation of KB.
- Usefulness of `wft` nodes:
  - Functional term with more than two arguments (slots).
  - Functional term with more than one filler in a slot.
  - Functional term shown as argument of another (filler in a slot).
- Can show a binary relation with no arc coming into it as a labeled arc ("collapsed arc").

Graph View as Visualization

- Visualized graph is for human comprehension.
- Visualized graph need not be isomorphic to implementation of KB.
- Usefulness of `wft` nodes:
  - Functional term with more than two arguments (slots).
  - Functional term with more than one filler in a slot.
  - Functional term shown as argument of another (filler in a slot).
- Can show a binary relation with no arc coming into it as a labeled arc ("collapsed arc").

Graph View as Visualization

- Visualized graph is for human comprehension.
- Visualized graph need not be isomorphic to implementation of KB.
- Usefulness of `wft` nodes:
  - Functional term with more than two arguments (slots).
  - Functional term with more than one filler in a slot.
  - Functional term shown as argument of another (filler in a slot).
- Can show a binary relation with no arc coming into it as a labeled arc ("collapsed arc").

Graph View as Visualization

- Visualized graph is for human comprehension.
- Visualized graph need not be isomorphic to implementation of KB.
- Usefulness of `wft` nodes:
  - Functional term with more than two arguments (slots).
  - Functional term with more than one filler in a slot.
  - Functional term shown as argument of another (filler in a slot).
- Can show a binary relation with no arc coming into it as a labeled arc ("collapsed arc").

Graph View as Visualization

- Visualized graph is for human comprehension.
- Visualized graph need not be isomorphic to implementation of KB.
- Usefulness of `wft` nodes:
  - Functional term with more than two arguments (slots).
  - Functional term with more than one filler in a slot.
  - Functional term shown as argument of another (filler in a slot).
- Can show a binary relation with no arc coming into it as a labeled arc ("collapsed arc").

Visualizing a Collapsed Arc

- Slots in a frame are ordered.
- Order of slots = order of arguments of functional term.
- Draw collapsed arc from first argument to second argument.
- Name of caseframe = function symbol.
- Label collapsed arc with function symbol.
- Different style of arrow head
  so user knows it's a collapsed arc.

Visualizing a Collapsed Arc

- Slots in a frame are ordered.
- Order of slots = order of arguments of functional term.
- Draw collapsed arc from first argument to second argument.
- Name of caseframe = function symbol.
- Label collapsed arc with function symbol.
- Different style of arrow head
  so user knows it's a collapsed arc.

Visualizing a Collapsed Arc

- Slots in a frame are ordered.
- Order of slots = order of arguments of functional term.
- Draw collapsed arc from first argument to second argument.
- Name of caseframe = function symbol.
- Label collapsed arc with function symbol.
- Different style of arrow head
  so user knows it's a collapsed arc.

Visualizing a Collapsed Arc

- Slots in a frame are ordered.
- Order of slots = order of arguments of functional term.
- Draw collapsed arc from first argument to second argument.
- Name of caseframe = function symbol.
- Label collapsed arc with function symbol.
- Different style of arrow head
  so user knows it's a collapsed arc.

Visualizing a Collapsed Arc

- Slots in a frame are ordered.
- Order of slots = order of arguments of functional term.
- Draw collapsed arc from first argument to second argument.

- Name of caseframe = function symbol.
- Label collapsed arc with function symbol.

- Different style of arrow head
  so user knows it's a collapsed arc.

Visualizing a Collapsed Arc

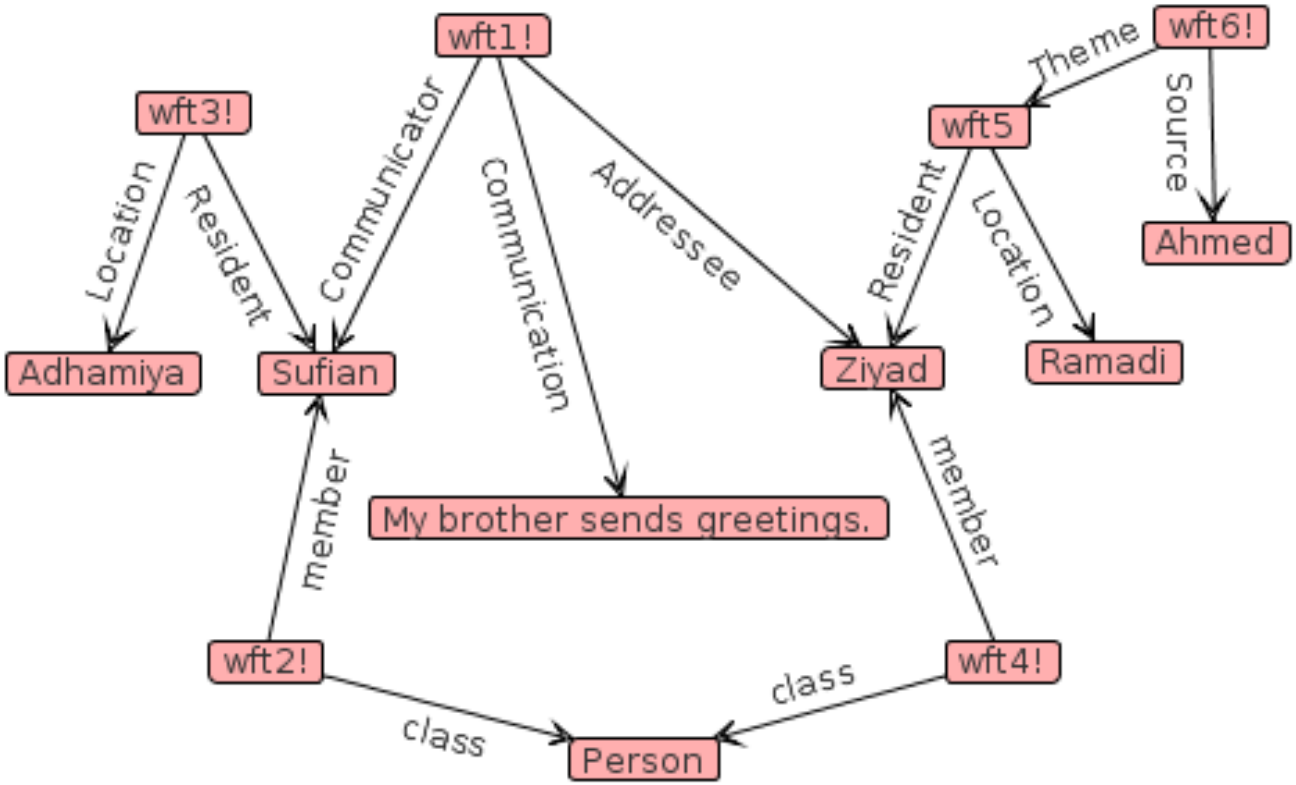
- Slots in a frame are ordered.
- Order of slots = order of arguments of functional term.
- Draw collapsed arc from first argument to second argument.

- Name of caseframe = function symbol.
- Label collapsed arc with function symbol.

- Different style of arrow head
  so user knows it's a collapsed arc.

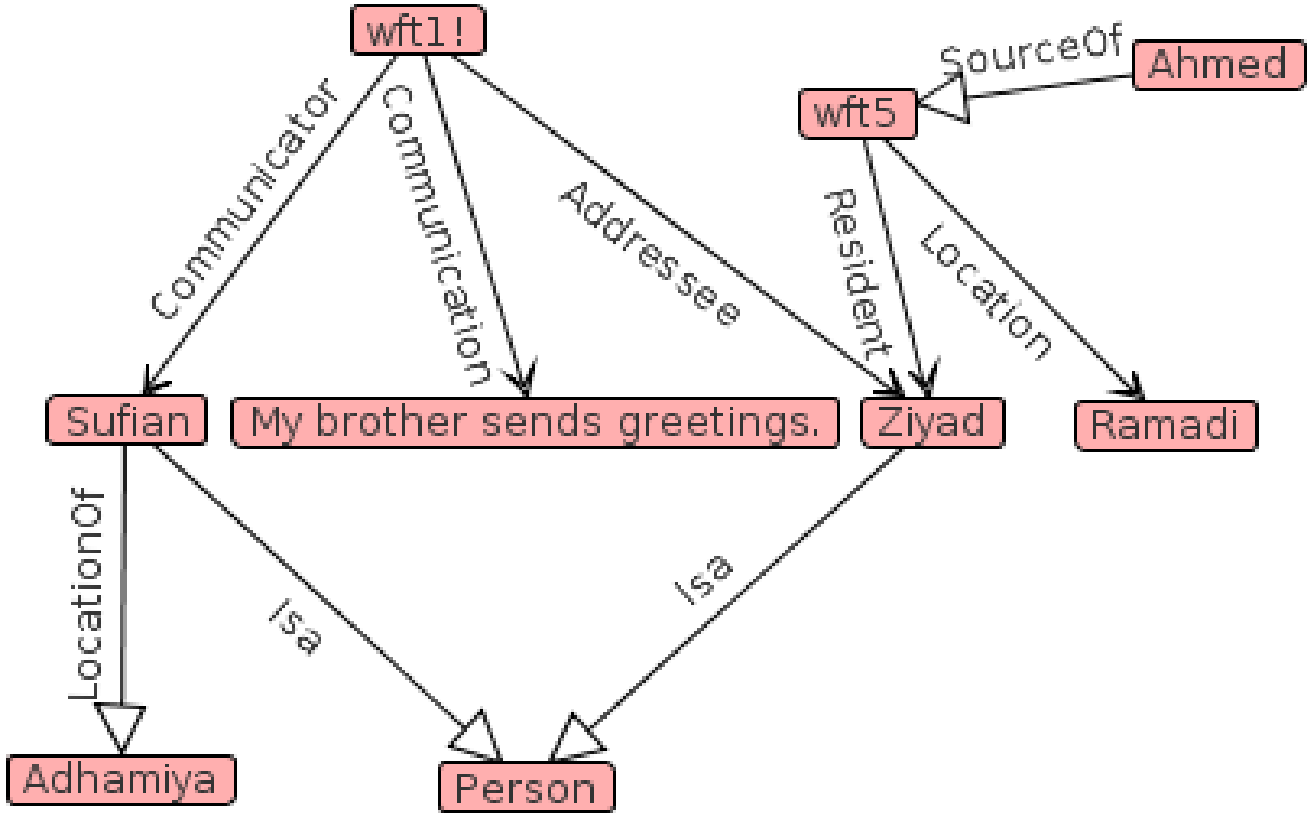
Example of Collapsed Graph: Before

The uncollapsed version of Suifian calling Ziyad example:

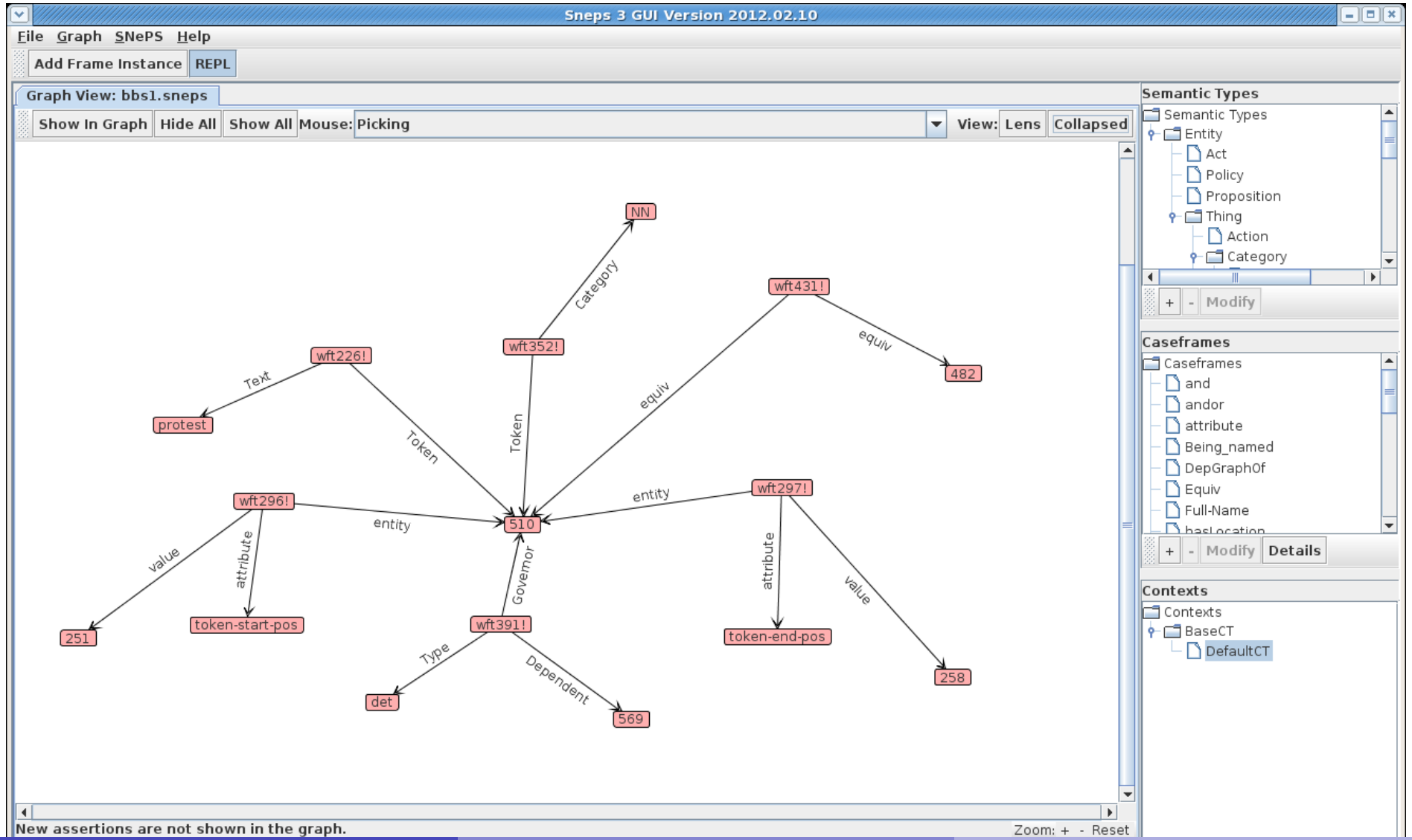


Example of Collapsed Graph: After

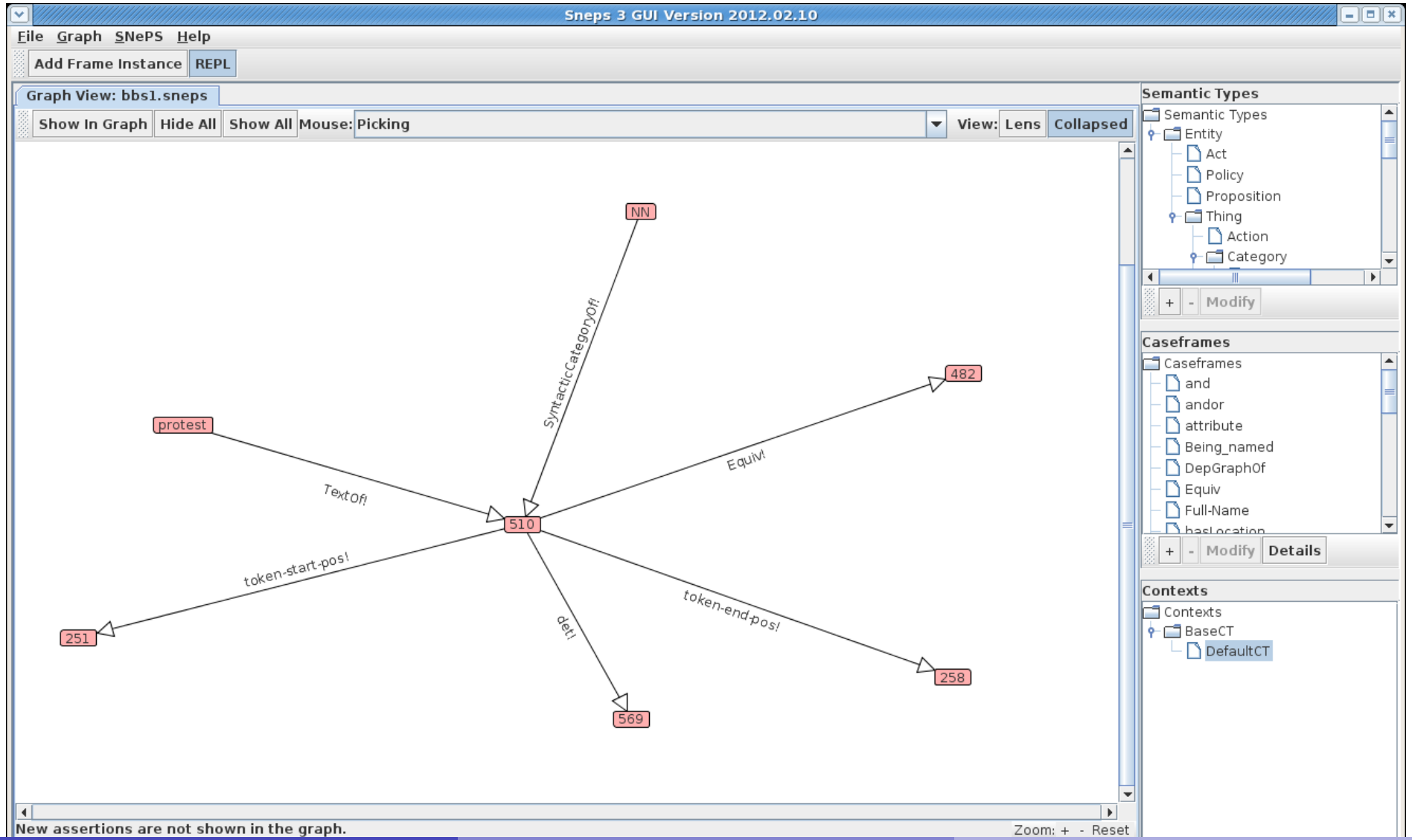
The collapsed version of Suifian calling Ziyad example:



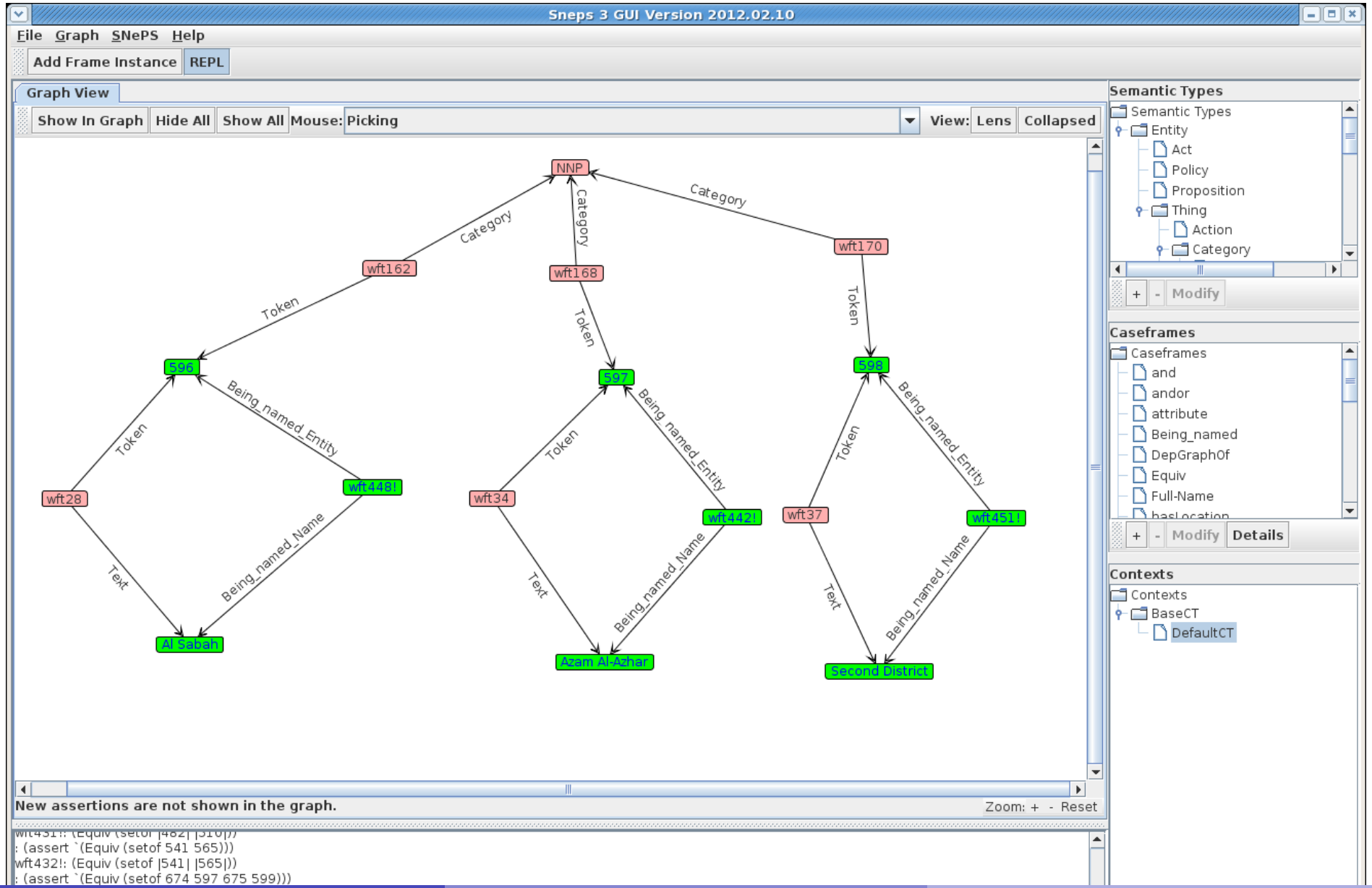
SNePS 3 Syntactic Graph Visualized (Uncollapsed)



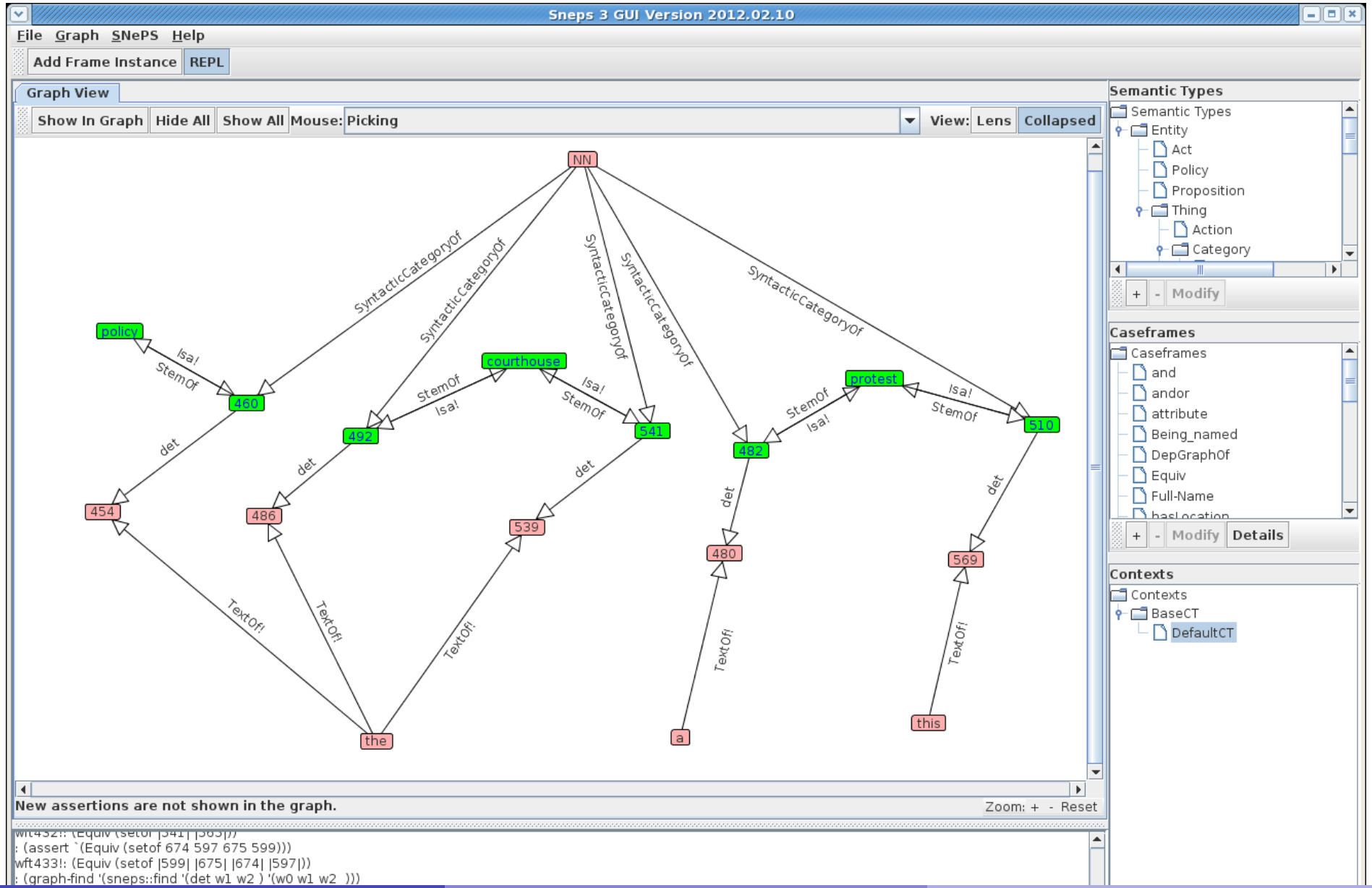
SNePS 3 Syntactic Graph Visualized (Collapsed)



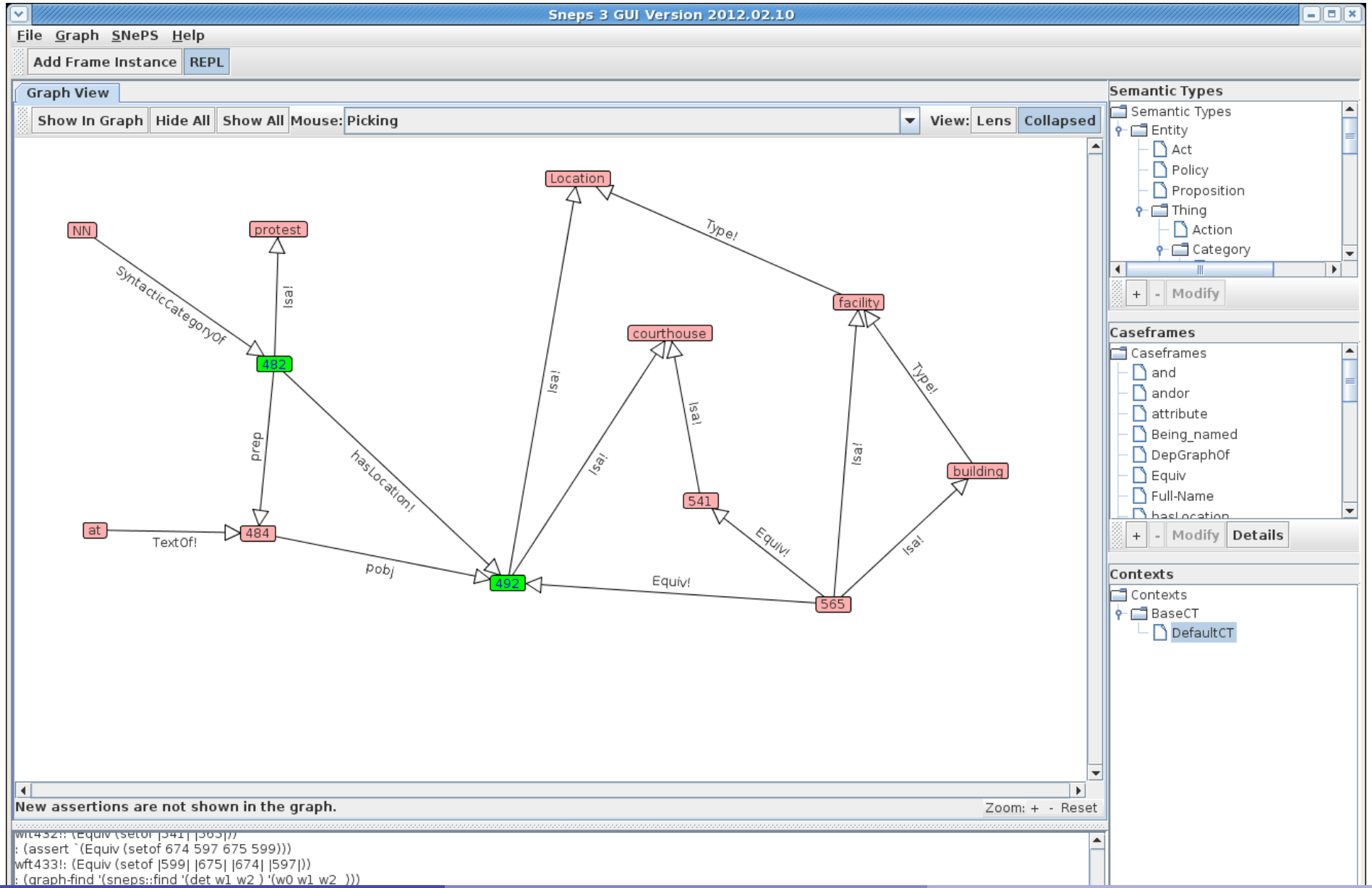
Mapping Rule: properNounToName



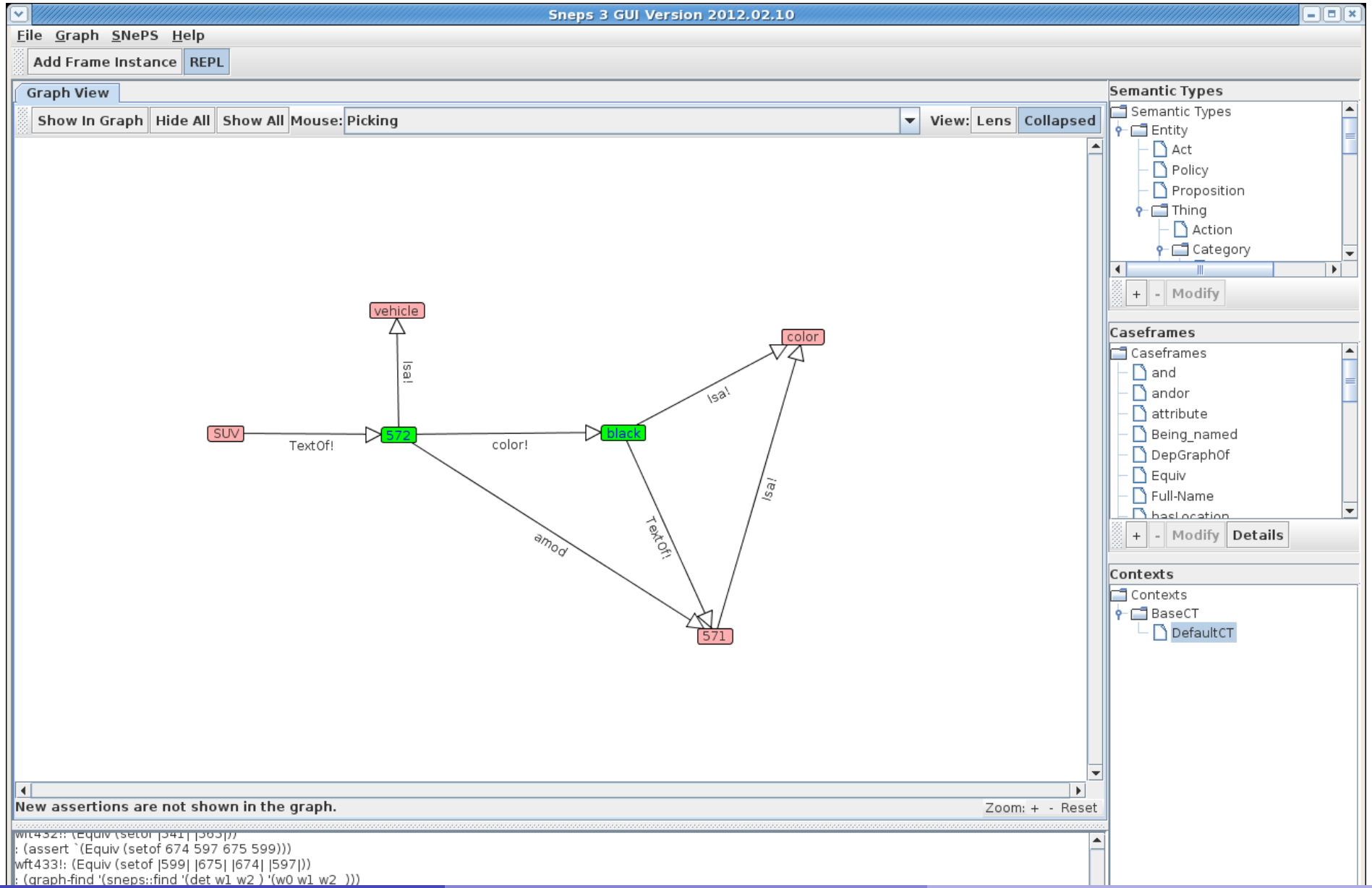
Mapping Rule: nounPhraseToInstance



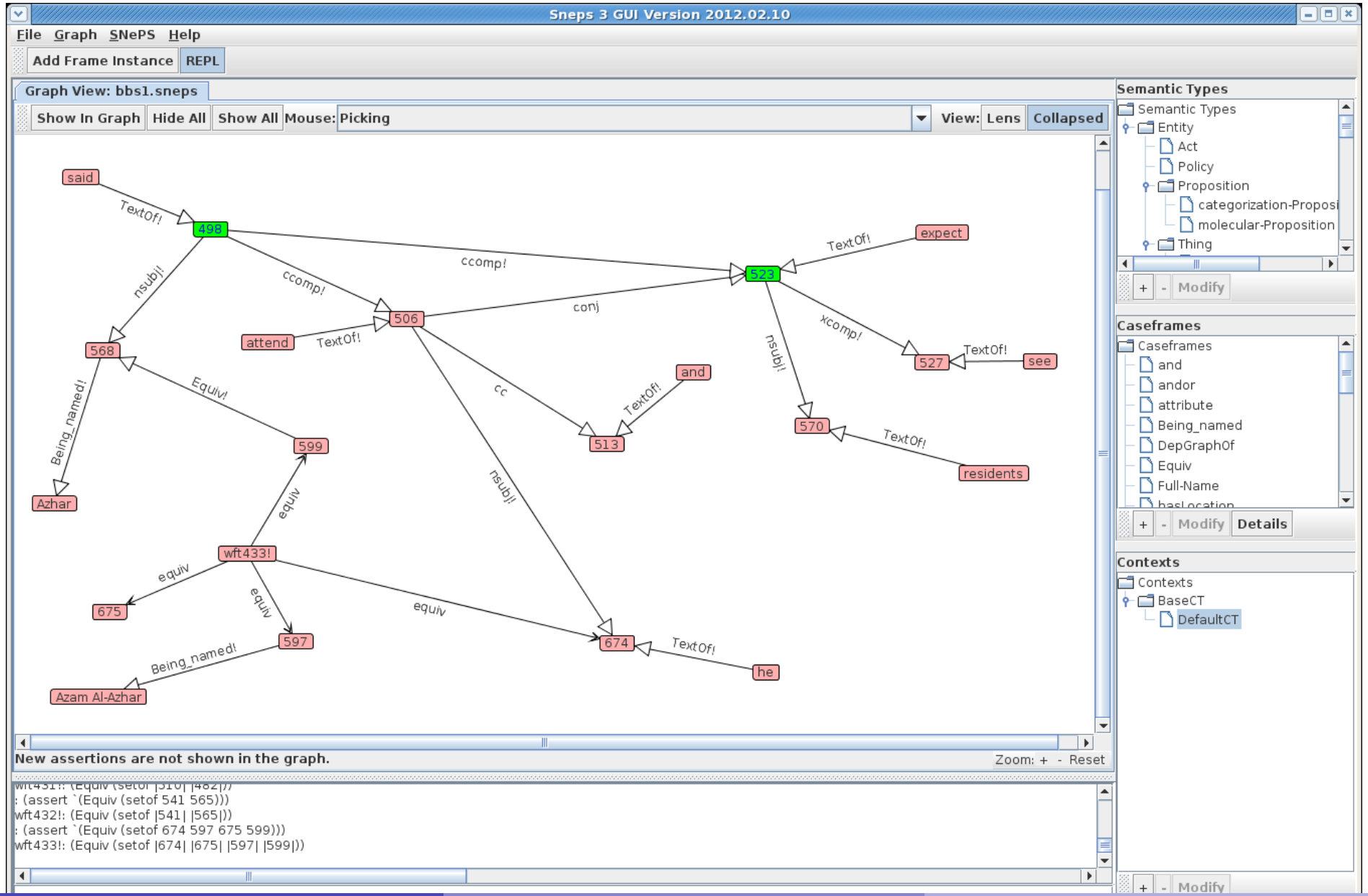
Mapping Rule: atLocation



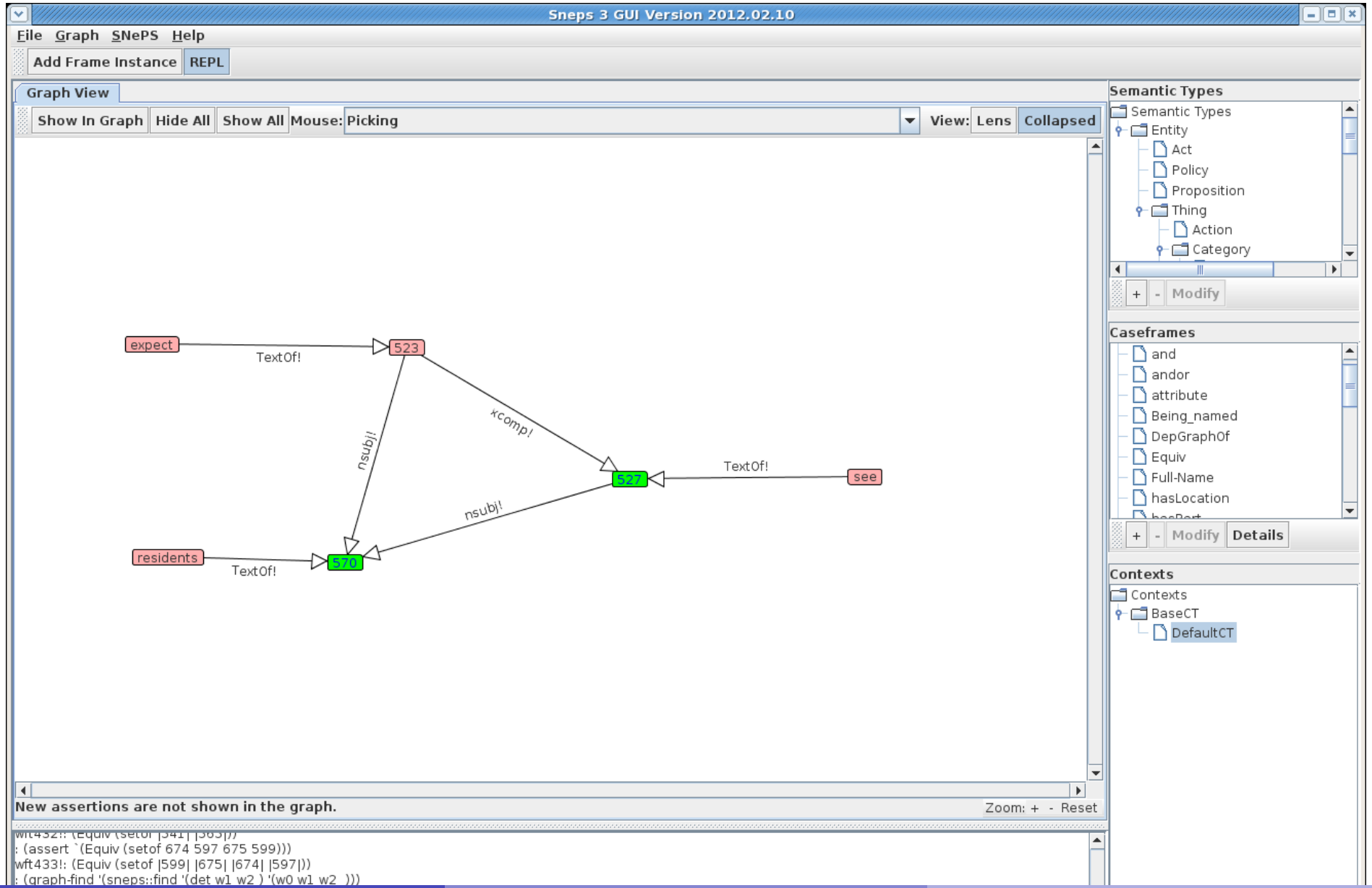
Mapping Rule: colorProperty



Mapping Rule: distribVerbOverConj



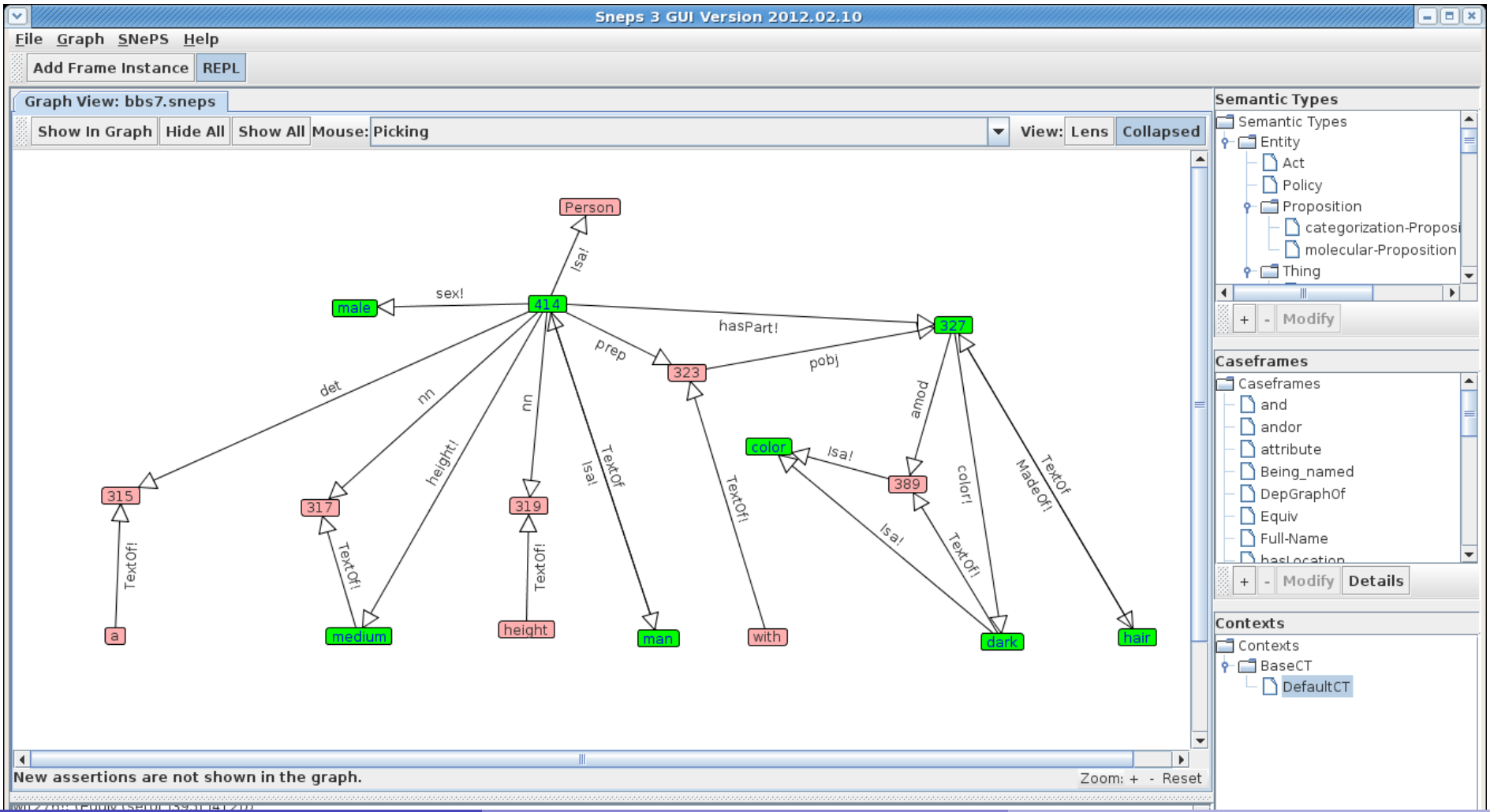
Mapping Rule: distribSubjOverXcomp



Another Example

7. 01/31/2010, 1817 hrs. – Blue team reports that
a medium height man with dark hair
just entered a blue car by the Second District Courthouse. The man was wearing a tan jacket. They are not sure where he came from.

Mapping Rules: nounPhraseToInstance; madeOfSubstance; colorProperty; hasDimensionValue; withTypicalPart



CBIR Productivity

Over the 7 messages of the Bomber Buster data set

Noun types looked up:	80
Noun tokens:	165
Assertions added to graphs:	11,860

Mapping Rule Usage

Rule	bbs1	bbs2	bbs3	bbs4	bbs5	bbs6	bbs7	Total
addRelevantOntology	1	1	1	1	1	1	2	8
substringCoreference	10	4	12	2	6	6	6	46
distribModOverConj	1	0	1	1	1	0	0	4
properNounToName	13	7	15	3	6	4	4	52
nounPhraseToInstance	5	9	4	12	4	2	5	41
madeOfSubstance	0	0	0	0	0	0	1	1
atLocation	1	0	0	0	1	1	0	3
modLocation	0	1	0	0	0	0	0	1
colorProperty	1	2	2	3	2	0	3	13
hasDimensionValue	0	1	0	0	0	0	1	2
withTypicalPart	0	0	0	0	0	0	1	1
distribVerbOverConj	1	0	1	0	1	0	0	3
passiveToActive	0	0	0	0	0	0	0	0
distribSubjOverXcomp	2	0	0	1	0	0	0	3
isaIsEquiv	0	2	0	1	0	0	0	3
prepToRelation	1	3	0	3	0	0	0	7
removeRedundantStems	56	59	37	73	28	17	40	310
removeTextWhenIsa	6	12	4	13	4	2	5	46
removeTextWhenMadeOf	0	0	0	0	0	0	1	1
Total	98	101	77	113	54	33	69	545

Timings

Average running time per message
Over the 7 messages of the Bomber Buster data set
On a Core i7 2600K @ 3.4ghz, with 16GB RAM

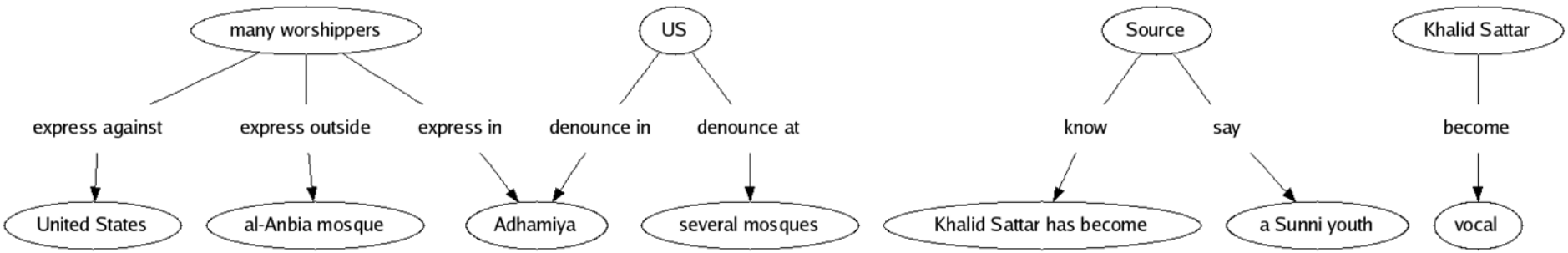
GATE (without human coreference editing):	2.642 secs
xml2sneps3:	0.978 secs
Propositionalizer (without CBIR):	0.596 secs
Propositionalizer (with CBIR):	74.547 secs

(Significant start-up time, so steady-state is faster.)

Motivational Example: STEF Messages

- 01/05/07 - Increased hostile sentiment being expressed against U.S. troops by many worshippers outside the al-Anbia mosque in Adhamiya.
- 01/06/07 - Source said a Sunni youth he knows, Khalid Sattar, has become increasingly vocal in denouncing the U.S. at several mosques in Adhamiya.

Motivational Example: STEF Graphs

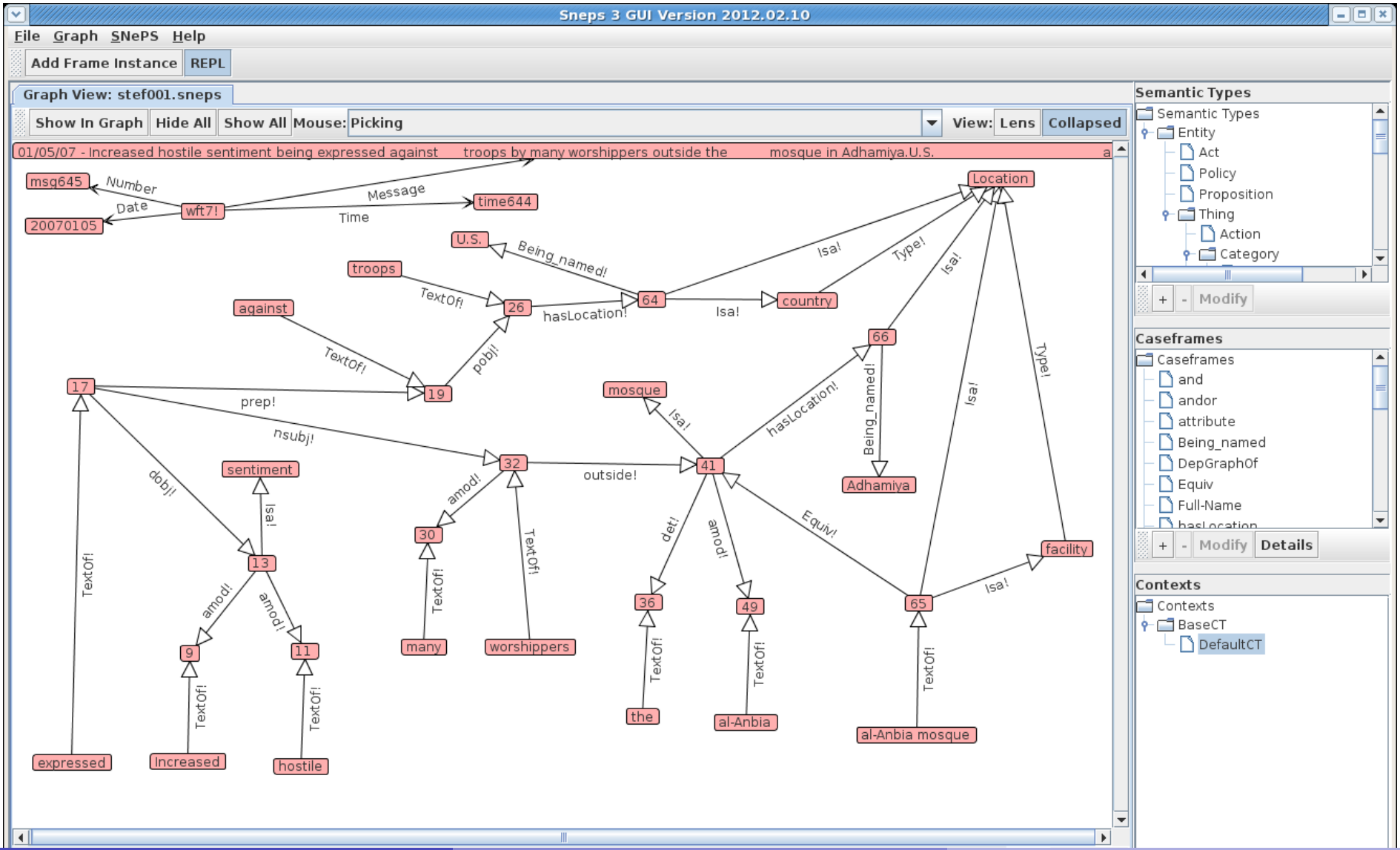


[K. Sambhoos, J. Llinas, & E. Little, Graphical Methods for Real-Time Fusion and Estimation with Soft Message Data, 11th International Conference on Information Fusion, 2008.]

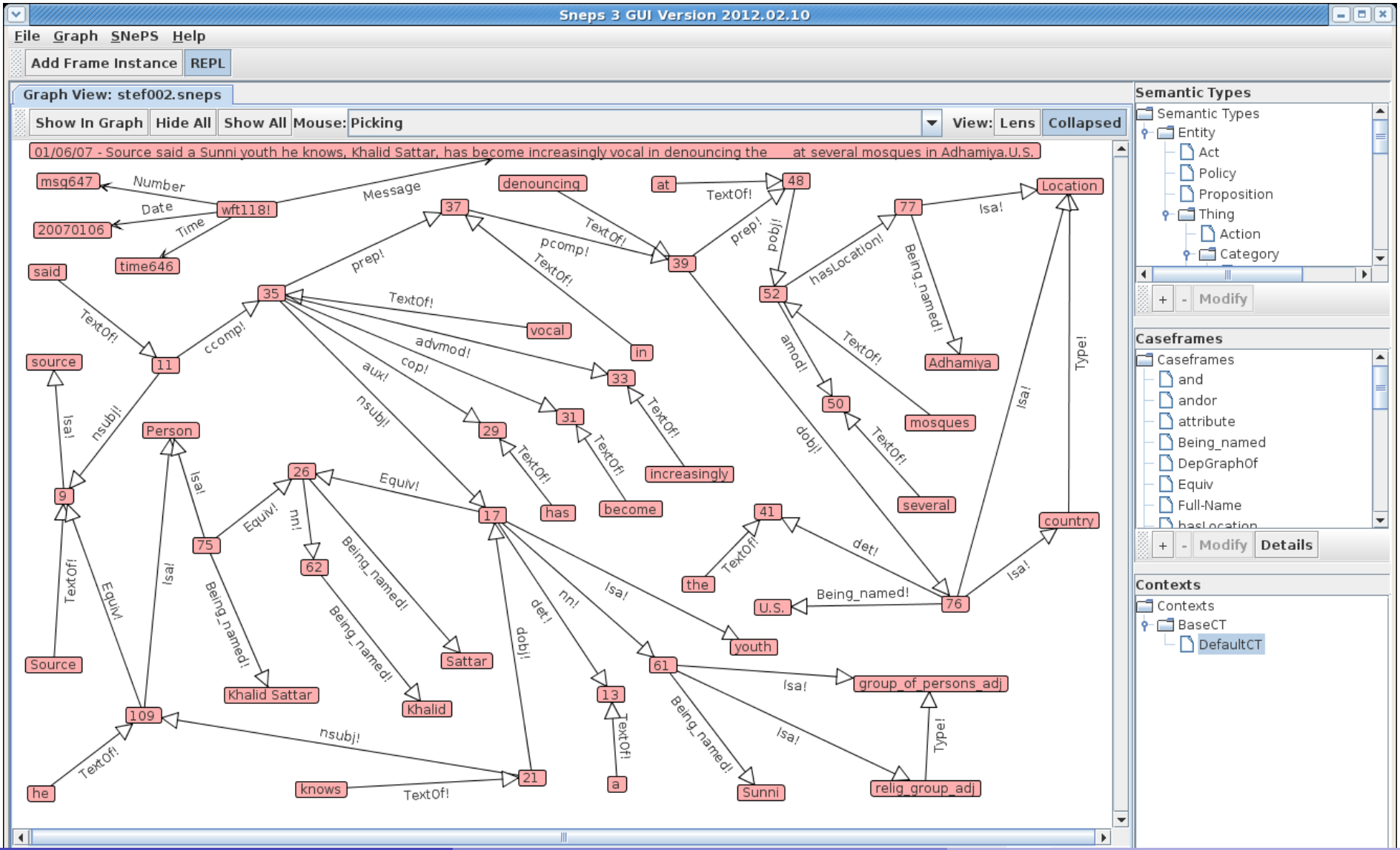
Questions:

- Relation between “US” and “United States”?
- What was expressed?
- Who denounced whom?
- Does Source know Khalid Sattar?
- Is Khalid Sattar a Sunni youth?
- What did Source say?
- When did these events occur?

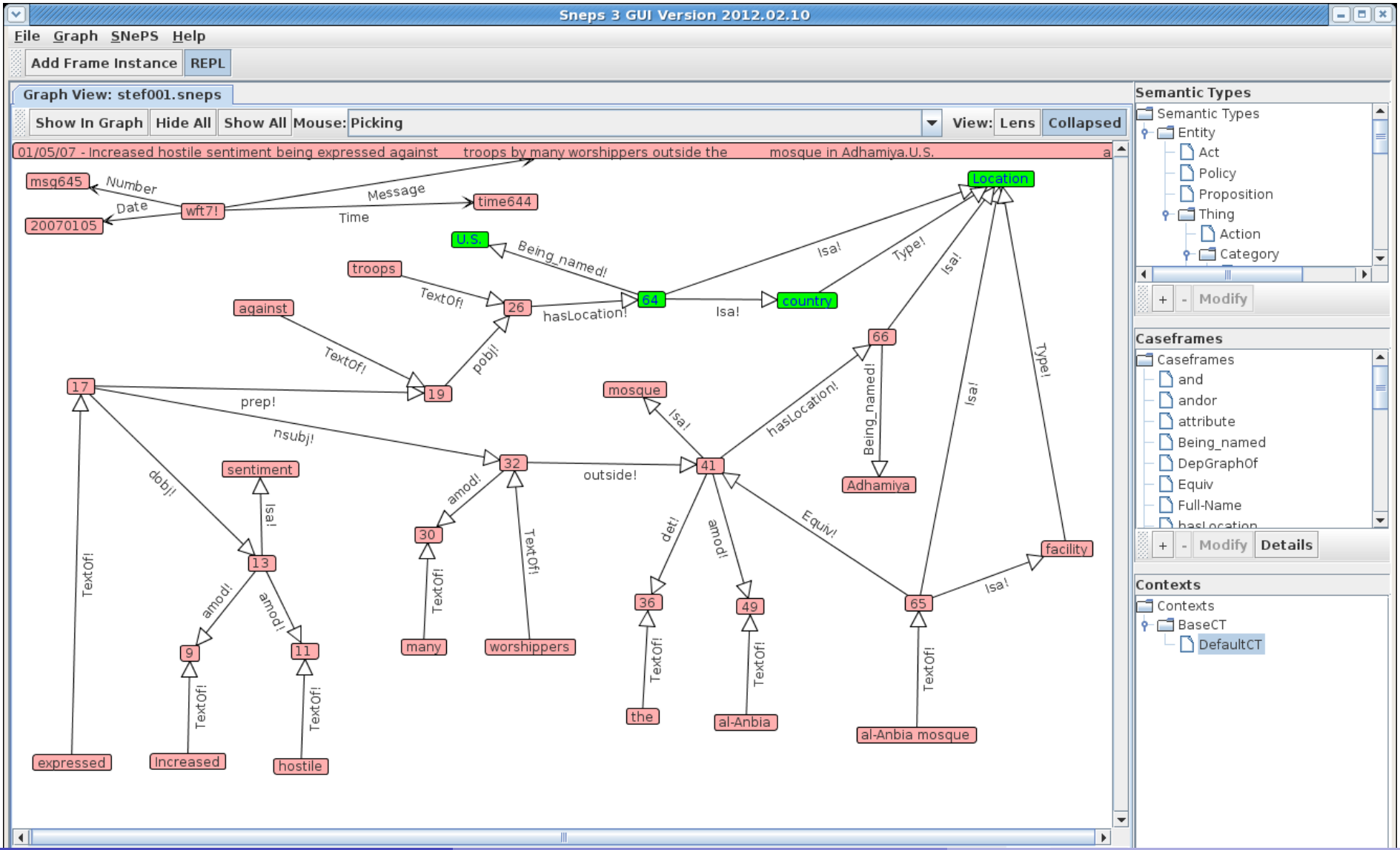
Tractor on STEF001



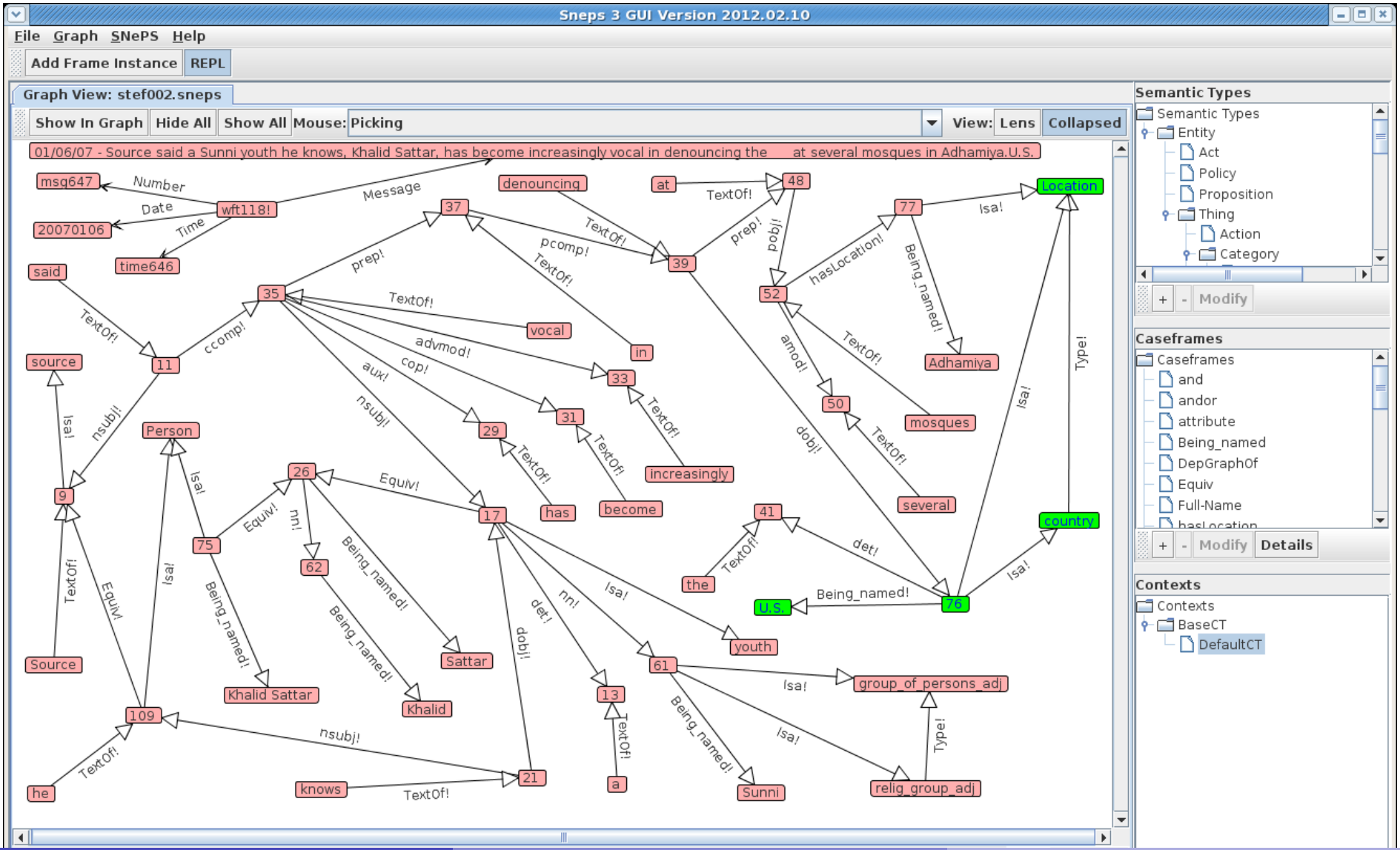
Tractor on STEF002



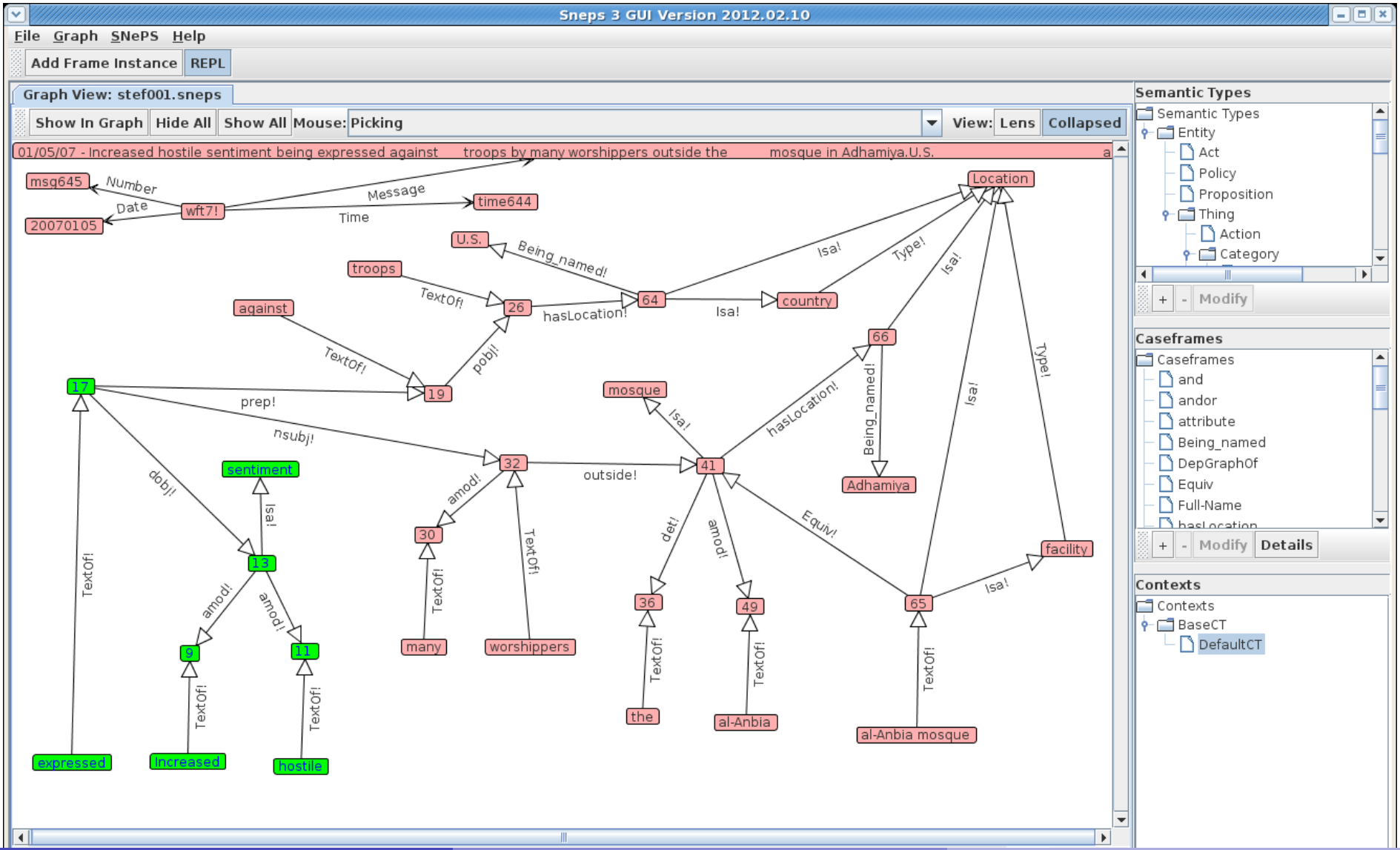
STEF001: “US”


Sneps 3 GUI Version 2012.02.10
File Graph SNePS Help
Add Frame Instance
REPL
Graph View: stef001.sneps
Show In Graph
Hide All
Show All
Mouse: Picking
View: Lens
Collapsed
01/05/07 - Increased hostile sentiment being expressed against troops by many worshippers outside the mosque in Adhamiya.U.S.
msg645
Number
Date
20070105
wft7!
Message
Time
time644
U.S.
Being_named!
troops
TextOf!
26
hasLocation!
64
Isa!
country
Location
Type!
against
TextOf!
19
pobj!
17
prep!
nsubj!
mosque
Isa!
hasLocation!
66
Being_named!
Adhamiya
32
outside!
41
sentiment
dobj!
Isa!
13
amod!
30
TextOf!
many
worshippers
det!
amod!
Equiv!
36
49
65
facility
9
11
TextOf!
expressed
Increased
hostile
the
al-Anbia
al-Anbia mosque
Semantic Types
Entity
Act
Policy
Proposition
Thing
Action
Category
Modify
Caseframes
and
andor
attribute
Being_named
DepGraphOf
Equiv
Full-Name
Modify
Details
Contexts
BaseCT
DefaultCT

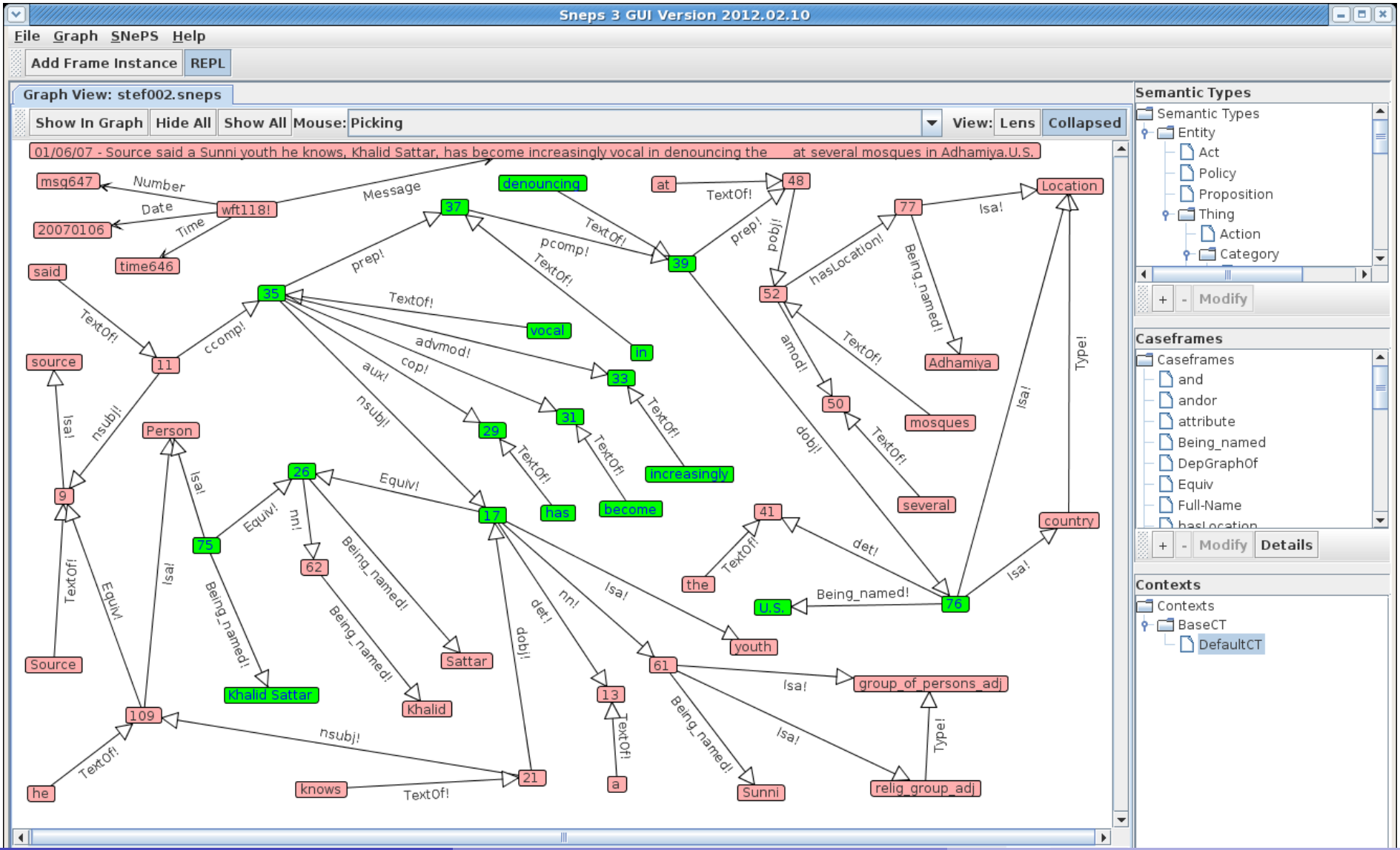
STEF002: “US”


Sneps 3 GUI Version 2012.02.10
File Graph SNePS Help
Add Frame Instance
REPL
Graph View: stef002.sneps
Show In Graph
Hide All
Show All
Mouse: Picking
View: Lens
Collapsed
01/06/07 - Source said a Sunni youth he knows, Khalid Sattar, has become increasingly vocal in denouncing the at several mosques in Adhamiya.U.S.
Semantic Types
Entity
Act
Policy
Proposition
Thing
Action
Category
Modify
Caseframes
and
andor
attribute
Being_named
DepGraphOf
Equiv
Full-Name
Modify
Details
Contexts
BaseCT
DefaultCT

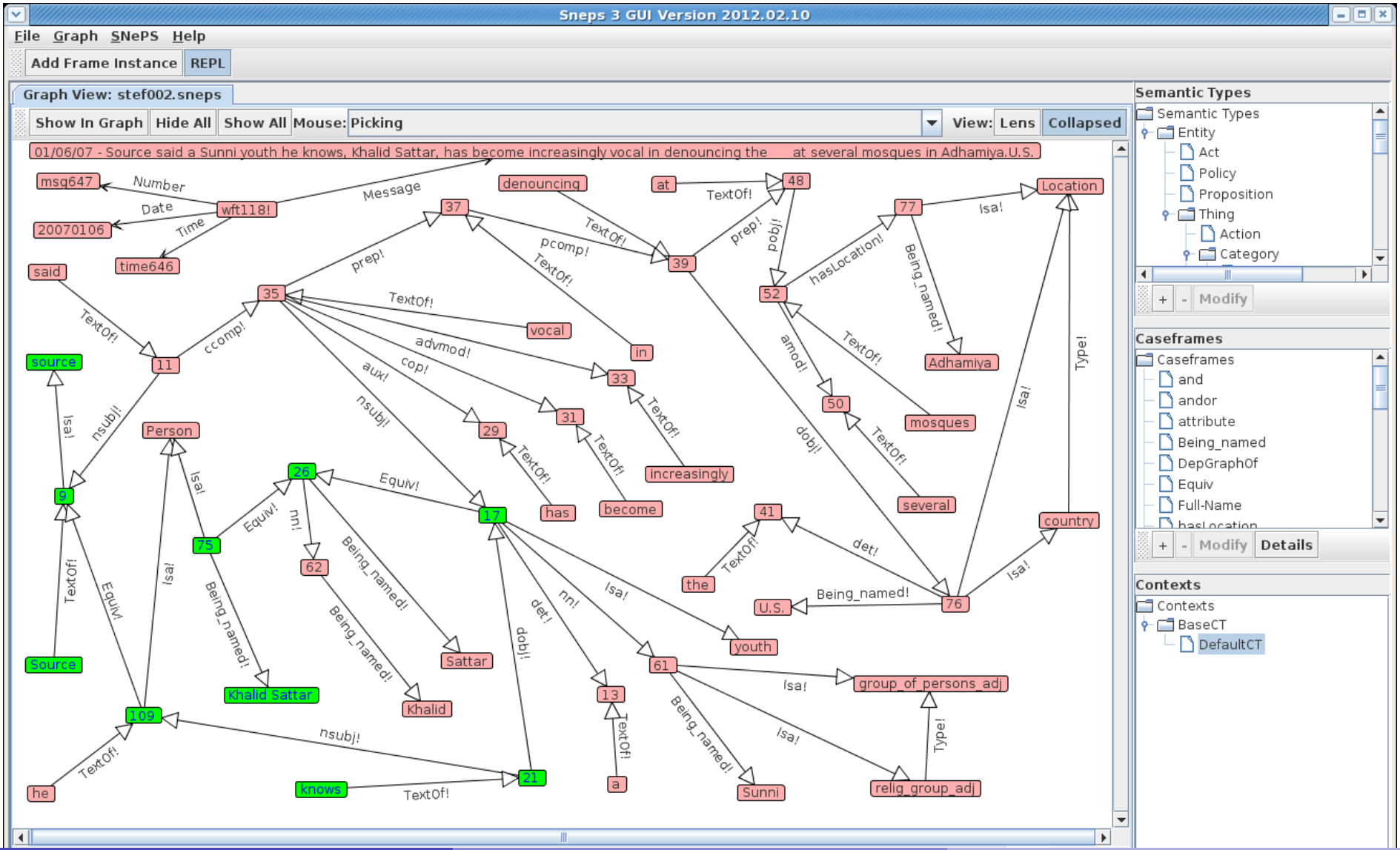
STEF001: What was Expressed?



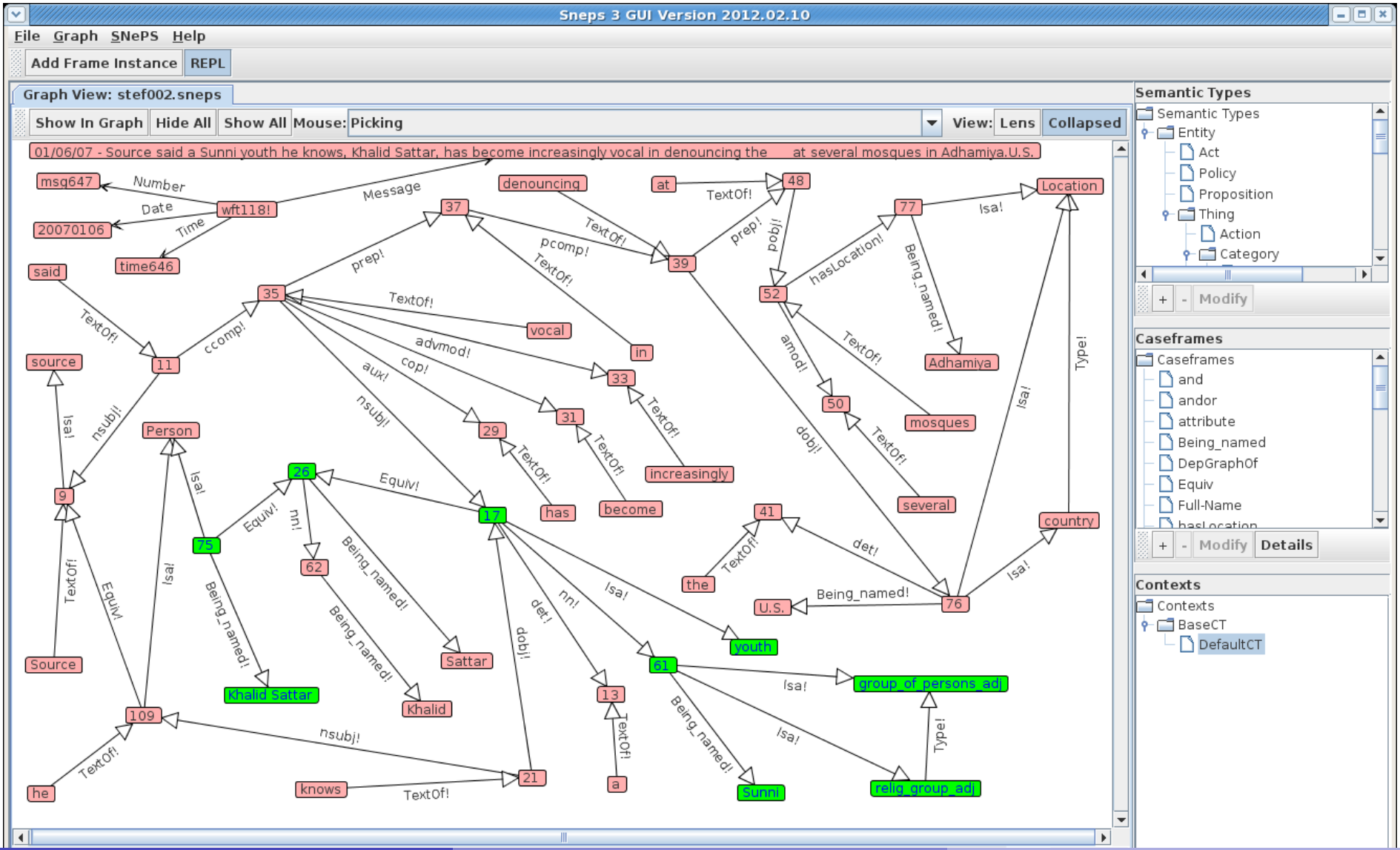
STEF002: Who denounced whom?



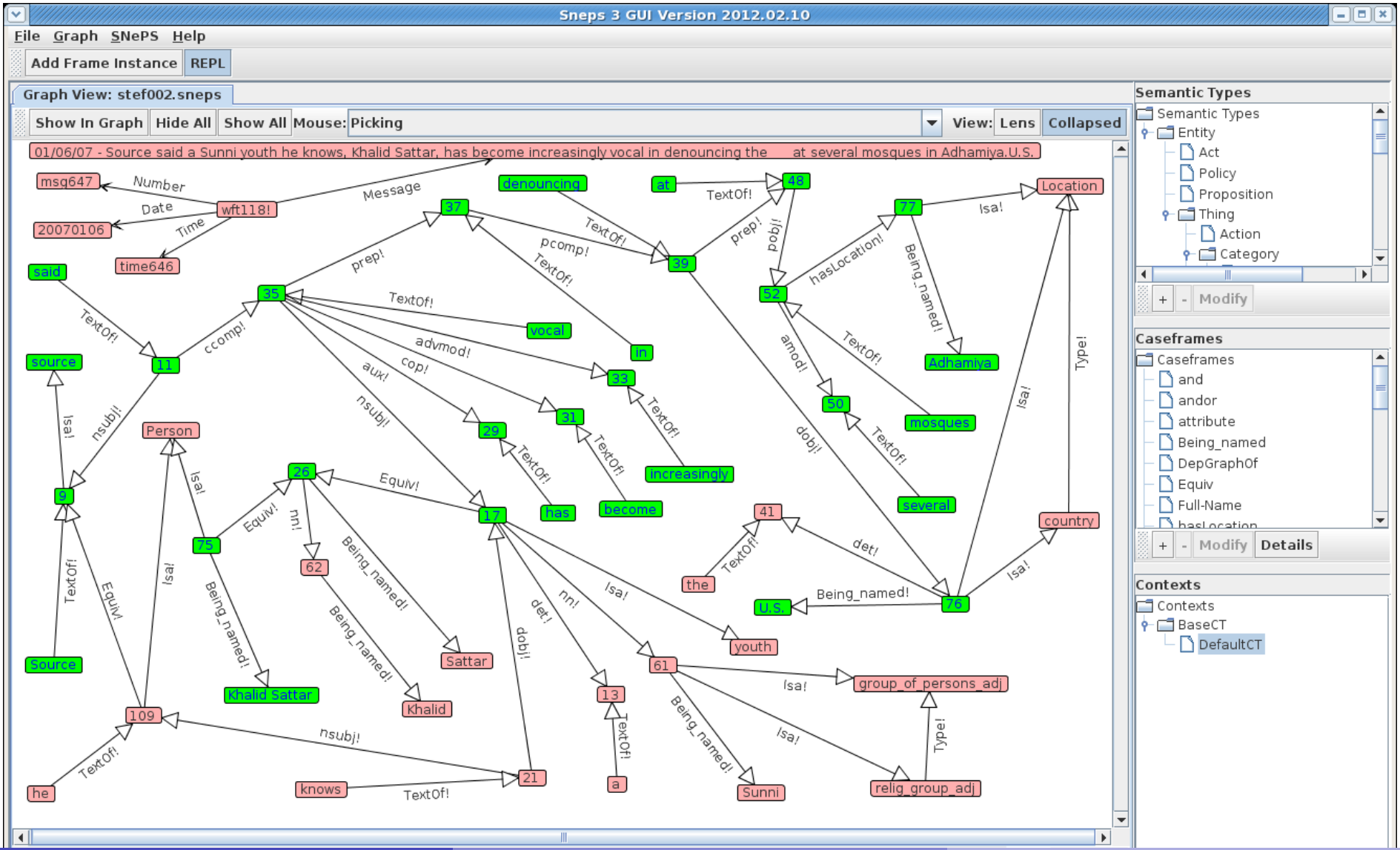
STEF002: Does Source know Khalid Sattar?



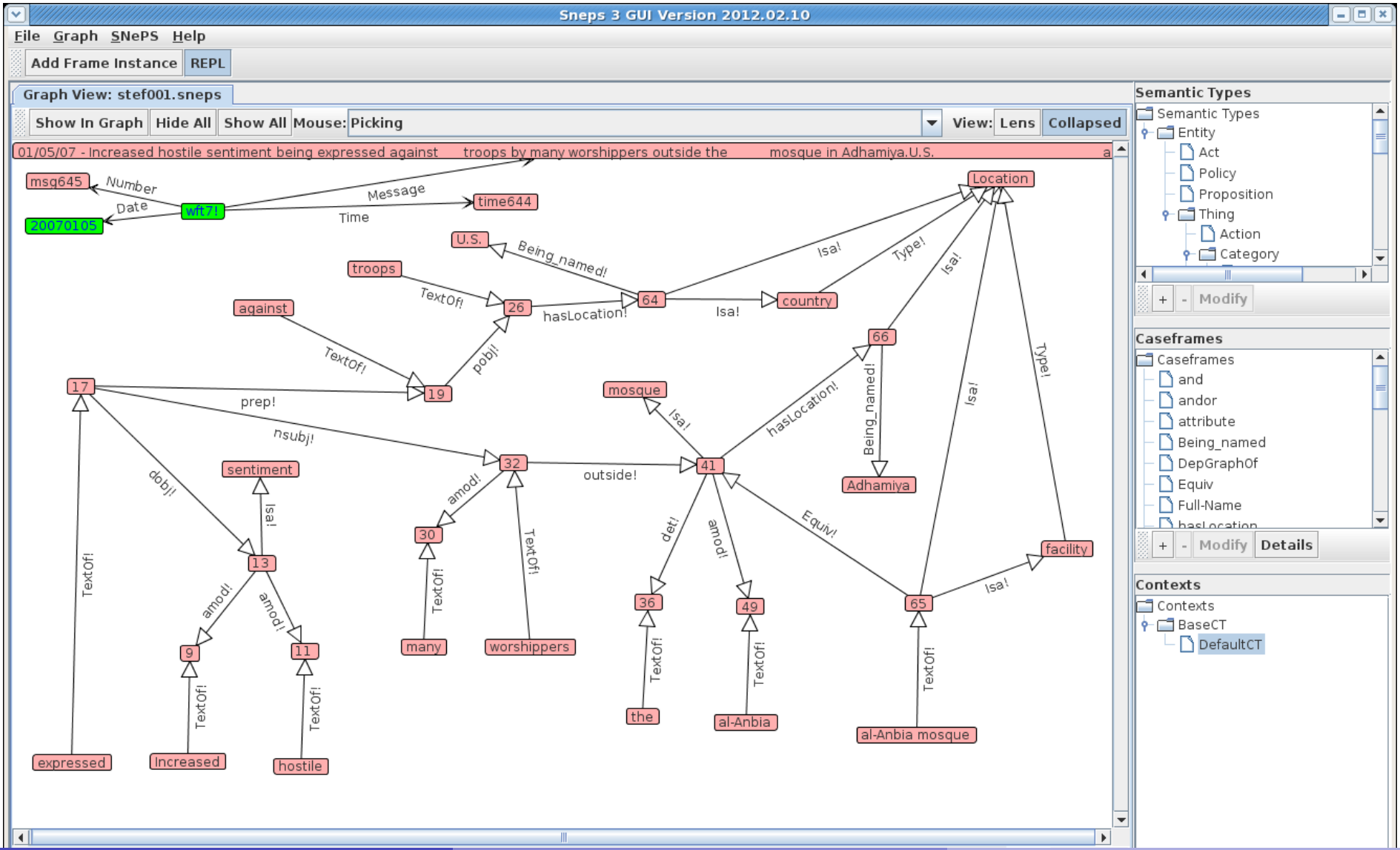
STEF002: Is Khalid Sattar a Sunni youth?



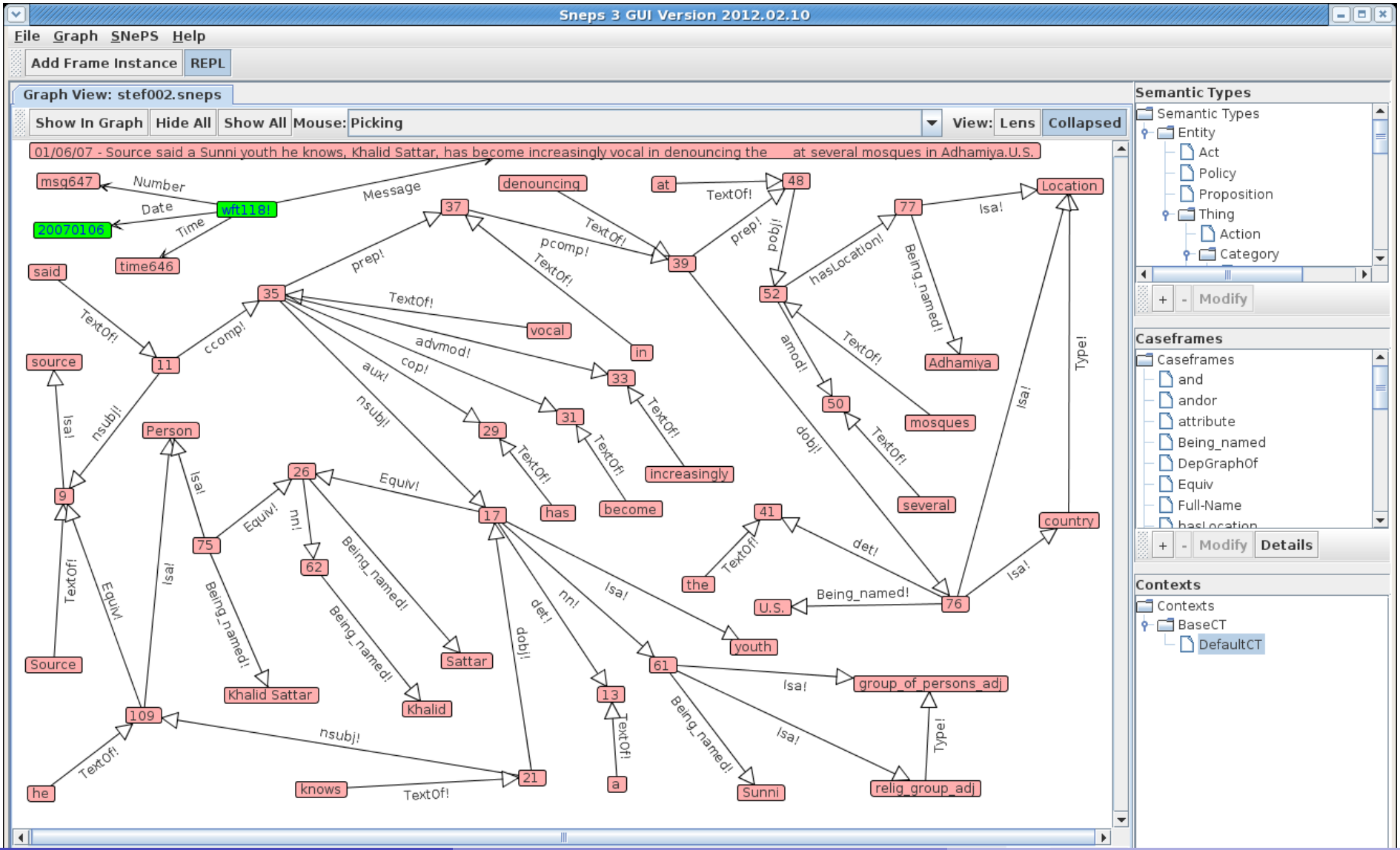
STEF002: What did Source say?



STEF001: When did these events occur?



STEF002: When did these events occur?



Summary

- Short English message, not necessarily “grammatical”.
- Syntactic processing within GATE.
- Named Entity Recognition, List-Based and Rule-Based.
- Automatic Coreferencers.
- Human Coreference Editing.
- Dependency Parser.
- Syntactic Propositional Graph: Syntactic Relations.
- Syntax-Semantics Mapping Rules.
- Semantic Propositional Graph reflects information in message.

Summary

- Short English message, not necessarily “grammatical”.
- Syntactic processing within GATE.
- Named Entity Recognition, List-Based and Rule-Based.
- Automatic Coreferencers.
- Human Coreference Editing.
- Dependency Parser.
- Syntactic Propositional Graph: Syntactic Relations.
- Syntax-Semantics Mapping Rules.
- Semantic Propositional Graph reflects information in message.

Summary

- Short English message, not necessarily “grammatical”.
- Syntactic processing within GATE.
- Named Entity Recognition, List-Based and Rule-Based.
- Automatic Coreferencers.
- Human Coreference Editing.
- Dependency Parser.
- Syntactic Propositional Graph: Syntactic Relations.
- Syntax-Semantics Mapping Rules.
- Semantic Propositional Graph reflects information in message.

Summary

- Short English message, not necessarily “grammatical”.
- Syntactic processing within GATE.
- Named Entity Recognition, List-Based and Rule-Based.
- Automatic Coreferencers.
- Human Coreference Editing.
- Dependency Parser.
- Syntactic Propositional Graph: Syntactic Relations.
- Syntax-Semantics Mapping Rules.
- Semantic Propositional Graph reflects information in message.

Summary

- Short English message, not necessarily “grammatical”.
- Syntactic processing within GATE.
- Named Entity Recognition, List-Based and Rule-Based.
- Automatic Coreferencers.
- Human Coreference Editing.
- Dependency Parser.
- Syntactic Propositional Graph: Syntactic Relations.
- Syntax-Semantics Mapping Rules.
- Semantic Propositional Graph reflects information in message.

Summary

- Short English message, not necessarily “grammatical”.
- Syntactic processing within GATE.
- Named Entity Recognition, List-Based and Rule-Based.
- Automatic Coreferencers.
- Human Coreference Editing.
- Dependency Parser.
- Syntactic Propositional Graph: Syntactic Relations.
- Syntax-Semantics Mapping Rules.
- Semantic Propositional Graph reflects information in message.

Summary

- Short English message, not necessarily “grammatical”.
- Syntactic processing within GATE.
- Named Entity Recognition, List-Based and Rule-Based.
- Automatic Coreferencers.
- Human Coreference Editing.
- Dependency Parser.
- Syntactic Propositional Graph: Syntactic Relations.
- Syntax-Semantics Mapping Rules.
- Semantic Propositional Graph reflects information in message.

Summary

- Short English message, not necessarily “grammatical”.
- Syntactic processing within GATE.
- Named Entity Recognition, List-Based and Rule-Based.
- Automatic Coreferencers.
- Human Coreference Editing.
- Dependency Parser.
- Syntactic Propositional Graph: Syntactic Relations.
- Syntax-Semantics Mapping Rules.
- Semantic Propositional Graph reflects information in message.

Summary

- Short English message, not necessarily “grammatical”.
- Syntactic processing within GATE.
- Named Entity Recognition, List-Based and Rule-Based.
- Automatic Coreferencers.
- Human Coreference Editing.
- Dependency Parser.
- Syntactic Propositional Graph: Syntactic Relations.
- Syntax-Semantics Mapping Rules.
- Semantic Propositional Graph reflects information in message.

Further Reading

- Michael Prentice, Michael Kandefer, & Stuart C. Shapiro, Tractor: A Framework for Soft Information Fusion, *Proceedings of the 13th International Conference on Information Fusion (Fusion 2010)*, 2010, Th3.2.2, 8 pages, unpaginated.

- Michael Prentice and Stuart C. Shapiro, Using Propositional Graphs for Soft Information Fusion, *Proceedings of the 14th International Conference on Information Fusion (Fusion 2011)*, 2011, 522–528.

- Daniel R. Schlegel and Stuart C. Shapiro, Visually Interacting with a Knowledge Base Using Frames, Logic, and Propositional Graphs. In M. Croitoru, S. Rudolph, N. Wilson, J. Howse, and O. Corby, Eds., *Graph Structures for Knowledge Representation and Reasoning, Lecture Notes in Artificial Intelligence 7205*, Springer-Verlag, Berlin, 2012, in press.