Propositional Knowledge Representation for Biomedical Informatics

Stuart C. Shapiro, PhD [1] Daniel R. Schlegel, PhD [2]

[1] Professor Emeritus, UB Department of Computer Science and Engineering
Affiliated Professor Emeritus, UB Department of Linguistics
Affiliated Professor Emeritus, UB Department of Philosophy
ACM Distinguished Scientist
Fellow, AAAI
shapiro@buffalo.edu

[2] Postdoctoral Associate
UB Department of Biomedical Informatics
drschlegel@buffalo.edu

Goals of Talk

1. Provide an appreciation of Knowledge Representation systems.
2. Discuss benefits of propositional representation.
3. Introduce the SNePS knowledge representation system.

Outline

1. Introduction: Knowledge Representation and Reasoning
2. Focus of Representation: Objects, Classes, Propositions
3. View of Representation: Logic, Frames, Graphs
   - The Logic View
   - The Frame View
   - The Graph View
4. SNePS
   - The Logic View of SNePS
   - The Frame View of SNePS
   - The Propositional Graph View of SNePS
5. Benefits of Propositional Knowledge Representation
6. Conclusions

Outline

Knowledge Representation

A well-defined declarative formalism
for expressing information
to be used by a computational system.

Well-defined: Having a clear syntax and semantics.

Clear syntax: A clear specification of atomic symbols and rules for combining them into expressions.

Clear semantics: Clear meanings of atomic symbols and clear rules for determining the meanings of expressions.

Declarative: Each expression can independently be

- entered by a user;
- produced by the system in response to a user query.

Knowledge Representation

A well-defined declarative formalism
for expressing information
to be used by a computational system.

Well-defined: Having a clear syntax and semantics.

Clear syntax: A clear specification of atomic symbols and rules for combining them into expressions.

Clear semantics: Clear meanings of atomic symbols and clear rules for determining the meanings of expressions.

Declarative: Each expression can independently be

- entered by a user;
- produced by the system in response to a user query.

Knowledge Representation

A well-defined declarative formalism
for expressing information
to be used by a computational system.

Well-defined: Having a clear syntax and semantics.

Clear syntax: A clear specification of atomic symbols and rules for combining them into expressions.

Clear semantics: Clear meanings of atomic symbols and clear rules for determining the meanings of expressions.

Declarative: Each expression can independently be

- entered by a user;
- produced by the system in response to a user query.

Knowledge Representation

A well-defined declarative formalism
for expressing information
to be used by a computational system.

Well-defined: Having a clear syntax and semantics.

Clear syntax: A clear specification of atomic symbols and rules for combining them into expressions.

Clear semantics: Clear meanings of atomic symbols and clear rules for determining the meanings of expressions.

Declarative: Each expression can independently be

- entered by a user;
- produced by the system in response to a user query.

Knowledge Representation

A well-defined declarative formalism
for expressing information
to be used by a computational system.

Well-defined: Having a clear syntax and semantics.

Clear syntax: A clear specification of atomic symbols and rules for combining them into expressions.

Clear semantics: Clear meanings of atomic symbols and clear rules for determining the meanings of expressions.

Declarative: Each expression can independently be

- entered by a user;
- produced by the system in response to a user query.

Reasoning

The process of
producing
and using
from the stored KR expressions
additional KR expressions
that are consequences of them.

Domain of Discourse

The "world" within which
the entities denoted by terms
(what the terms mean)
reside.

Could be:
the real world;
the world of some theoretical discipline;
a fictional or mythological world.

Knowledge Base (KB)

All the expressions of some KR system accessible by some computational system at some time.

User

A person supplying or obtaining information from the system.

(We are KR system designers/implementers.
I am agnostic regarding (most) issues of content.)

Outline

Objects vs. Classes vs. Propositions

Objects: The individuals/instances in the domain of discourse.
Dr. Jan's patient, Jane Doe.

Classes: Classes/categories of objects,
sometimes called "concepts".
The class of patients.

Propositions: Facts or beliefs about objects or classes.

Jane Doe is a patient.
Patients are persons.
Jane Doe has a prescription of metformin from Dr. Jan.

Objects vs. Classes vs. Propositions

Objects: The individuals/instances in the domain of discourse.
Dr. Jan's patient, Jane Doe.

Classes: Classes/categories of objects,
sometimes called "concepts".
The class of patients.

Propositions: Facts or beliefs about objects or classes.

Jane Doe is a patient.
Patients are persons.
Jane Doe has a prescription of metformin from Dr. Jan.

Objects vs. Classes vs. Propositions

Objects: The individuals/instances in the domain of discourse.
Dr. Jan's patient, Jane Doe.

Classes: Classes/categories of objects,
sometimes called "concepts".
The class of patients.

Propositions: Facts or beliefs about objects or classes.

Jane Doe is a patient.
Patients are persons.
Jane Doe has a prescription of metformin from Dr. Jan.

What is a Proposition?

An expression in some language

- that can be true or false
- whose negation makes sense
- that can be believed or not
- whose negation can be believed or not
- that can be put in the frame
  "I believe that it is not the case that _____."

Sentences vs. Propositions

A sentence is an expression of a (written natural) language
that begins with a capital letter
and ends with a period, question mark, or exclamation point.

Some sentences do not contain a proposition:
"Hi!", "Why?", "Pass the salt!"

Some sentences do not express a proposition, but contain one:
"Does Jane have diabetes?"

Some sentences contain more than one proposition:
If Jane has diabetes, then Jane should take metformin.

Sentences vs. Propositions

A sentence is an expression of a (written natural) language
that begins with a capital letter
and ends with a period, question mark, or exclamation point.

Some sentences do not contain a proposition:
"Hi!", "Why?", "Pass the salt!"

Some sentences do not express a proposition, but contain one:
"Does Jane have diabetes?"

Some sentences contain more than one proposition:
If Jane has diabetes, then Jane should take metformin.

Sentences vs. Propositions

A sentence is an expression of a (written natural) language
that begins with a capital letter
and ends with a period, question mark, or exclamation point.

Some sentences do not contain a proposition:
“Hi!”, “Why?”, “Pass the salt!”

Some sentences do not express a proposition, but contain one:
“Does Jane have diabetes?”

Some sentences contain more than one proposition:
If Jane has diabetes, then Jane should take metformin.

Sentences vs. Propositions

A sentence is an expression of a (written natural) language
that begins with a capital letter
and ends with a period, question mark, or exclamation point.

Some sentences do not contain a proposition:
“Hi!”, “Why?”, “Pass the salt!”

Some sentences do not express a proposition, but contain one:
“Does Jane have diabetes?”

Some sentences contain more than one proposition:
If Jane has diabetes, then Jane should take metformin.

Significance of Focus

What the user thinks about when giving and getting information.

What one can (easily) give and get information about.

The entities in the domain of discourse.

Significance of Focus

What the user thinks about when giving and getting information.

What one can (easily) give and get information about.

The entities in the domain of discourse.

Significance of Focus

What the user thinks about when giving and getting information.

What one can (easily) give and get information about.

The entities in the domain of discourse.

Focusing on Objects

Jane Doe

- is a patient.
- is 56 years old.
- is female.
- has a prescription of metformin from Dr. Jan.
- has a prescription of cisplatin from Dr. Park.

Focusing on Classes

Primary Care Physicians are Physicians
Clinical Oncologists are Physicians
Physicians are Persons
Patients are Persons

Physicians prescribe Treatments to Patients

Focusing on Propositions

Jane Doe is a Patient.
Dr. Jan is a Primary Care Physician.
Primary Care Physicians are Physicians.
Metformin is a Medication.

Jane Doe is female.

Jane Doe has a prescription of metformin from Dr. Jan.

Dr. Park reported that Jane Doe has lung cancer.

Jane Doe does not have arthritis.

Focusing on Propositions

Jane Doe is a Patient.
Dr. Jan is a Primary Care Physician.
Primary Care Physicians are Physicians.
Metformin is a Medication.

Jane Doe is female.

Jane Doe has a prescription of metformin from Dr. Jan.

Dr. Park reported that Jane Doe has lung cancer.

Jane Doe does not have arthritis.

Focusing on Propositions

Jane Doe is a Patient.
Dr. Jan is a Primary Care Physician.
Primary Care Physicians are Physicians.
Metformin is a Medication.

Jane Doe is female.

Jane Doe has a prescription of metformin from Dr. Jan.

Dr. Park reported that Jane Doe has lung cancer.

Jane Doe does not have arthritis.

Focusing on Propositions

Jane Doe is a Patient.
Dr. Jan is a Primary Care Physician.
Primary Care Physicians are Physicians.
Metformin is a Medication.

Jane Doe is female.

Jane Doe has a prescription of metformin from Dr. Jan.

Dr. Park reported that Jane Doe has lung cancer.

Jane Doe does not have arthritis.

Focusing on Propositions

Jane Doe is a Patient.
Dr. Jan is a Primary Care Physician.
Primary Care Physicians are Physicians.
Metformin is a Medication.

Jane Doe is female.

Jane Doe has a prescription of metformin from Dr. Jan.

Dr. Park reported that Jane Doe has lung cancer.

Jane Doe does not have arthritis.

Outline

A KB Could be

- A collection of logical expressions.
- A collection of frames.
- A graph.

Outline

What is a Logic?

- A language for expressing domain entities and propositions.
  (Call expressions that express propositions “sentences”.)
- Has operators to combine sentences into compound sentences.
- Allows a proposition to have a “belief” or “truth” status.
- Each operator specifies a way of determining
  the meaning of a compound sentence
  from the meanings of the combined sentences.
  (“Compositional semantics.”)
- A notion of logical consequence.
- A system for reasoning over propositions.
- Prevents reasoning from “truths” to “falsities”.
  (But can reason from truths and falsities to truths and falsities.)

What is a Logic?

- A language for expressing domain entities and propositions.
  (Call expressions that express propositions “sentences”.)
- Has operators to combine sentences into compound sentences.
- Allows a proposition to have a “belief” or “truth” status.
- Each operator specifies a way of determining
  the meaning of a compound sentence
  from the meanings of the combined sentences.
  (“Compositional semantics.”)
- A notion of logical consequence.
- A system for reasoning over propositions.
- Prevents reasoning from “truths” to “falsities”.
  (But can reason from truths and falsities to truths and falsities.)

What is a Logic?

- A language for expressing domain entities and propositions.
  (Call expressions that express propositions “sentences”.)
- Has operators to combine sentences into compound sentences.
- Allows a proposition to have a “belief” or “truth” status.
- Each operator specifies a way of determining
  the meaning of a compound sentence
  from the meanings of the combined sentences.
  (“Compositional semantics.”)
- A notion of logical consequence.
- A system for reasoning over propositions.
- Prevents reasoning from “truths” to “falsities”.
  (But can reason from truths and falsities to truths and falsities.)

What is a Logic?

- A language for expressing domain entities and propositions.
  (Call expressions that express propositions “sentences”.)
- Has operators to combine sentences into compound sentences.
- Allows a proposition to have a “belief” or “truth” status.
- Each operator specifies a way of determining
  the meaning of a compound sentence
  from the meanings of the combined sentences.
  (“Compositional semantics.”)
- A notion of logical consequence.
- A system for reasoning over propositions.
- Prevents reasoning from “truths” to “falsities”.
  (But can reason from truths and falsities to truths and falsities.)

What is a Logic?

- A language for expressing domain entities and propositions.
  (Call expressions that express propositions “sentences”.)
- Has operators to combine sentences into compound sentences.
- Allows a proposition to have a “belief” or “truth” status.
- Each operator specifies a way of determining
  the meaning of a compound sentence
  from the meanings of the combined sentences.
  (“Compositional semantics.”)
- A notion of logical consequence.
- A system for reasoning over propositions.
- Prevents reasoning from “truths” to “falsities”.
  (But can reason from truths and falsities to truths and falsities.)

What is a Logic?

- A language for expressing domain entities and propositions.
  (Call expressions that express propositions “sentences”.)
- Has operators to combine sentences into compound sentences.
- Allows a proposition to have a “belief” or “truth” status.
- Each operator specifies a way of determining
  the meaning of a compound sentence
  from the meanings of the combined sentences.
  (“Compositional semantics.”)
- A notion of logical consequence.
- A system for reasoning over propositions.
- Prevents reasoning from “truths” to “falsities”.
  (But can reason from truths and falsities to truths and falsities.)

What is a Logic?

- A language for expressing domain entities and propositions.
  (Call expressions that express propositions “sentences”.)
- Has operators to combine sentences into compound sentences.
- Allows a proposition to have a “belief” or “truth” status.
- Each operator specifies a way of determining
  the meaning of a compound sentence
  from the meanings of the combined sentences.
  (“Compositional semantics.”)
- A notion of logical consequence.
- A system for reasoning over propositions.
- Prevents reasoning from “truths” to “falsities”.
  (But can reason from truths and falsities to truths and falsities.)

Outline

What is a Frame?

- A data structure in computer memory.
- Represents an entity in the domain.
- Has a set of named **slots**.
- Each slot represents a binary relation.
- The slots are not entities in the domain.
- Each slot is filled by one or more symbols or other frames.
- Frame_1

  Slot_i: Frame_2

  represents that the relation of Slot_i holds between the entity of Frame_1 and the entity of Frame_2.
- Computationally:
  Given Frame_1 and Slot_i, constant time access to Frame_2.

What is a Frame?

- A data structure in computer memory.
- Represents an entity in the domain.
- Has a set of named **slots**.
- Each slot represents a binary relation.
- The slots are not entities in the domain.
- Each slot is filled by one or more symbols or other frames.
- Frame_1

  Slot_i: Frame_2

  represents that the relation of Slot_i holds between the entity of Frame_1 and the entity of Frame_2.
- Computationally:
  Given Frame_1 and Slot_i, constant time access to Frame_2.

What is a Frame?

- A data structure in computer memory.
- Represents an entity in the domain.
- Has a set of named **slots**.
- Each slot represents a binary relation.
- The slots are not entities in the domain.
- Each slot is filled by one or more symbols or other frames.
- Frame_1

  Slot_i: Frame_2

  represents that the relation of Slot_i holds between the entity of Frame_1 and the entity of Frame_2.
- Computationally:
  Given Frame_1 and Slot_i, constant time access to Frame_2.

What is a Frame?

- A data structure in computer memory.
- Represents an entity in the domain.
- Has a set of named **slots**.
- Each slot represents a binary relation.
- The slots are not entities in the domain.
- Each slot is filled by one or more symbols or other frames.
- Frame_1

  Slot_i: Frame_2

  represents that the relation of Slot_i holds between the entity of Frame_1 and the entity of Frame_2.
- Computationally:
  Given Frame_1 and Slot_i, constant time access to Frame_2.

What is a Frame?

- A data structure in computer memory.
- Represents an entity in the domain.
- Has a set of named **slots**.
- Each slot represents a binary relation.
- The slots are not entities in the domain.
- Each slot is filled by one or more symbols or other frames.
- Frame_1

  Slot_i: Frame_2

  represents that the relation of Slot_i holds between the entity of Frame_1 and the entity of Frame_2.
- Computationally:
  Given Frame_1 and Slot_i, constant time access to Frame_2.

What is a Frame?

- A data structure in computer memory.
- Represents an entity in the domain.
- Has a set of named **slots**.
- Each slot represents a binary relation.
- The slots are not entities in the domain.
- Each slot is filled by one or more symbols or other frames.
- Frame_1
  Slot_i: Frame_2
  represents that the relation of Slot_i holds between the entity of Frame_1 and the entity of Frame_2.
- Computationally:
  Given Frame_1 and Slot_i, constant time access to Frame_2.

What is a Frame?

- A data structure in computer memory.
- Represents an entity in the domain.
- Has a set of named **slots**.
- Each slot represents a binary relation.
- The slots are not entities in the domain.
- Each slot is filled by one or more symbols or other frames.
- Frame_1
  Slot_i: Frame_2
  represents that the relation of Slot_i holds between the entity of Frame_1 and the entity of Frame_2.
- Computationally:
  Given Frame_1 and Slot_i, constant time access to Frame_2.

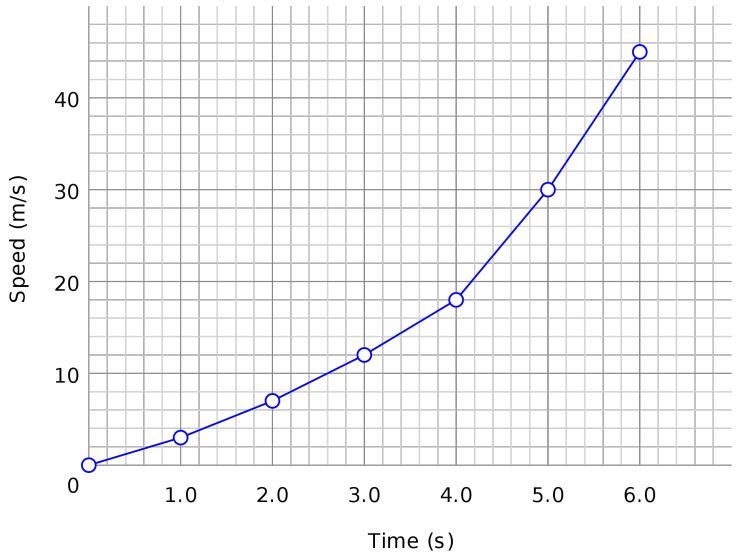
What is a Frame?

- A data structure in computer memory.
- Represents an entity in the domain.
- Has a set of named **slots**.
- Each slot represents a binary relation.
- The slots are not entities in the domain.
- Each slot is filled by one or more symbols or other frames.
- Frame_1
  Slot_i: Frame_2
  represents that the relation of Slot_i holds between the entity of Frame_1 and the entity of Frame_2.
- Computationally:
  Given Frame_1 and Slot_i, constant time access to Frame_2.

Outline

What is a Graph?

Analytic geometry:



Graph theory:

- A Mathematical Structure
- A Data Structure
- A Picture

A Graph as a Mathematical Structure

A Graph, G, is an ordered pair, $\langle V, E \rangle$ of:

1. V: a set of entities, called *vertices* or *nodes*;
2. E: a set of two-element subsets of V, called *edges*.

A Directed Graph, G, is an ordered pair, $\langle V, E \rangle$ of:

1. V: a set of entities, called *vertices* or *nodes*;
2. E: a set of ordered pairs, $\langle v_1, v_2 \rangle$, called *arcs*, where v_1 and v_2 are in V.

A Labeled Directed Graph, G, is an ordered pair, $\langle V, E \rangle$ of:

1. V: a set of entities, called *vertices* or *nodes*;
2. E: a set of ordered triples, called *arcs*, $\langle v_1, v_2, l \rangle$ where v_1 and v_2 are in V, and l is a symbol called a *label*.

A Tree is a graph such that there is exactly one path between any two vertices.

A Graph as a Mathematical Structure

A Graph, G, is an ordered pair, $\langle V, E \rangle$ of:

1. V: a set of entities, called *vertices* or *nodes*;
2. E: a set of two-element subsets of V, called *edges*.

A Directed Graph, G, is an ordered pair, $\langle V, E \rangle$ of:

1. V: a set of entities, called *vertices* or *nodes*;
2. E: a set of ordered pairs, $\langle v_1, v_2 \rangle$, called *arcs*, where v_1 and v_2 are in V.

A Labeled Directed Graph, G, is an ordered pair, $\langle V, E \rangle$ of:

1. V: a set of entities, called *vertices* or *nodes*;
2. E: a set of ordered triples, called *arcs*, $\langle v_1, v_2, l \rangle$ where v_1 and v_2 are in V, and l is a symbol called a *label*.

A Tree is a graph such that there is exactly one path between any two vertices.

A Graph as a Mathematical Structure

A Graph, G, is an ordered pair, $\langle V, E \rangle$ of:

1. V: a set of entities, called *vertices* or *nodes*;
2. E: a set of two-element subsets of V, called *edges*.

A Directed Graph, G, is an ordered pair, $\langle V, E \rangle$ of:

1. V: a set of entities, called *vertices* or *nodes*;
2. E: a set of ordered pairs, $\langle v_1, v_2 \rangle$, called *arcs*, where v_1 and v_2 are in V.

A Labeled Directed Graph, G, is an ordered pair, $\langle V, E \rangle$ of:

1. V: a set of entities, called *vertices* or *nodes*;
2. E: a set of ordered triples, called *arcs*, $\langle v_1, v_2, l \rangle$ where v_1 and v_2 are in V, and l is a symbol called a *label*.

A Tree is a graph such that there is exactly one path between any two vertices.

A Graph as a Mathematical Structure

A Graph, G, is an ordered pair, $\langle V, E \rangle$ of:

1. V: a set of entities, called *vertices* or *nodes*;
2. E: a set of two-element subsets of V, called *edges*.

A Directed Graph, G, is an ordered pair, $\langle V, E \rangle$ of:

1. V: a set of entities, called *vertices* or *nodes*;
2. E: a set of ordered pairs, $\langle v_1, v_2 \rangle$, called *arcs*, where v_1 and v_2 are in V.

A Labeled Directed Graph, G, is an ordered pair, $\langle V, E \rangle$ of:

1. V: a set of entities, called *vertices* or *nodes*;
2. E: a set of ordered triples, called *arcs*, $\langle v_1, v_2, l \rangle$ where v_1 and v_2 are in V, and l is a symbol called a *label*.

A Tree is a graph such that there is exactly one path between any two vertices.

A Graph as a Data Structure

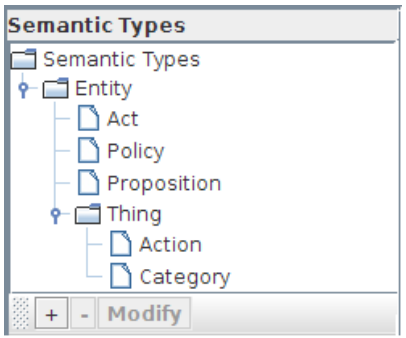
A (Labeled, Directed) Graph is

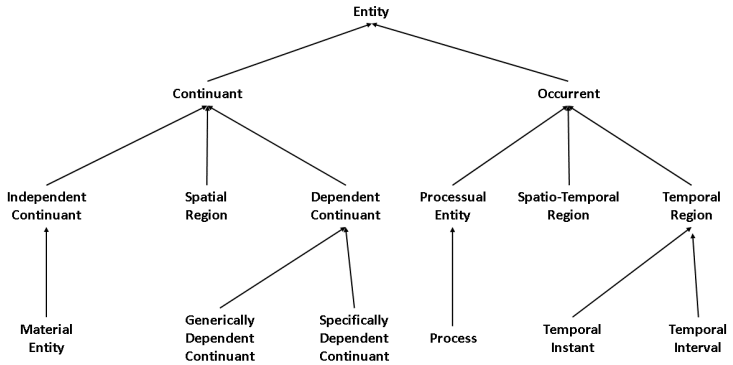
1. a set of data objects, called *vertices* or *nodes*;
2. a set of *labels*.

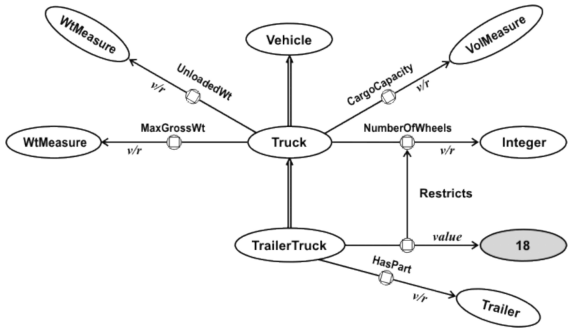
such that, given a vertex v_i and a label l_j, there is constant time access to a set of vertices V_{ij}.

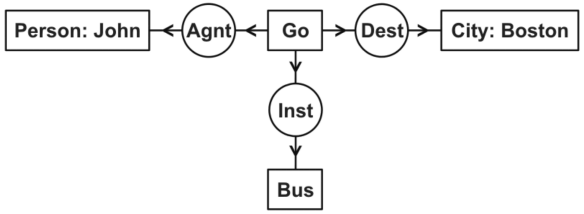
We say that an arc labeled l_j points from v_i to each vertex in V_{ij}.

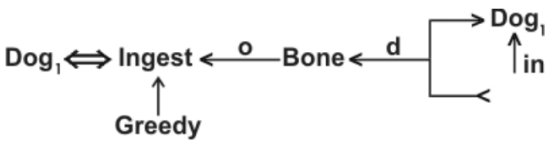
A Graph as a Picture, A Gallery of KR Picture Graphs

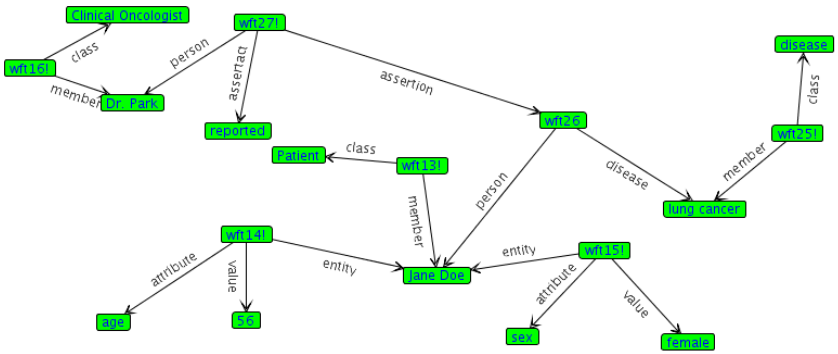


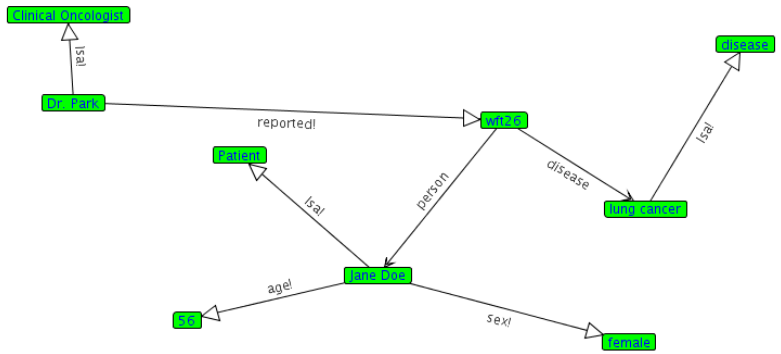












WARNING

Just because two picture graphs look different doesn't mean that they are implemented differently as data structures.

Just because two picture graphs look the same doesn't mean that they are implemented the same way as data structures.

Outline

Design Purpose

SNePS is a KRR system designed to
represent the beliefs of
a general, intelligent, natural-language-using, computational agent.

To represent anything an agent can conceive of.

Admit relations as entities in the domain.

Admit propositions as entities in the domain.

Allow propositions about any entity.

Uniqueness Principle: Anything represented is represented exactly once.

Therefore: Two different representations
represent different (in some way) entities.

Design Purpose

SNePS is a KRR system designed to
represent the beliefs of
a general, intelligent, natural-language-using, computational agent.

To represent anything an agent can conceive of.

Admit relations as entities in the domain.

Admit propositions as entities in the domain.

Allow propositions about any entity.

Uniqueness Principle: Anything represented is represented exactly once.

Therefore: Two different representations
represent different (in some way) entities.

Design Purpose

SNePS is a KRR system designed to
represent the beliefs of
a general, intelligent, natural-language-using, computational agent.

To represent anything an agent can conceive of.

Admit relations as entities in the domain.

Admit propositions as entities in the domain.

Allow propositions about any entity.

Uniqueness Principle: Anything represented is represented exactly once.

Therefore: Two different representations
represent different (in some way) entities.

Design Purpose

SNePS is a KRR system designed to
represent the beliefs of
a general, intelligent, natural-language-using, computational agent.

To represent anything an agent can conceive of.

Admit relations as entities in the domain.

Admit propositions as entities in the domain.

Allow propositions about any entity.

Uniqueness Principle: Anything represented is represented exactly once.

Therefore: Two different representations
represent different (in some way) entities.

Design Purpose

SNePS is a KRR system designed to
represent the beliefs of
a general, intelligent, natural-language-using, computational agent.

To represent anything an agent can conceive of.

Admit relations as entities in the domain.

Admit propositions as entities in the domain.

Allow propositions about any entity.

Uniqueness Principle: Anything represented is represented exactly once.

Therefore: Two different representations
represent different (in some way) entities.

Design Purpose

SNePS is a KRR system designed to
represent the beliefs of
a general, intelligent, natural-language-using, computational agent.

To represent anything an agent can conceive of.

Admit relations as entities in the domain.

Admit propositions as entities in the domain.

Allow propositions about any entity.

Uniqueness Principle: Anything represented is represented exactly once.

Therefore: Two different representations
represent different (in some way) entities.

Design Purpose

SNePS is a KRR system designed to
represent the beliefs of
a general, intelligent, natural-language-using, computational agent.

To represent anything an agent can conceive of.

Admit relations as entities in the domain.

Admit propositions as entities in the domain.

Allow propositions about any entity.

Uniqueness Principle: Anything represented is represented exactly once.

Therefore: Two different representations
represent different (in some way) entities.

CSNePS

CSNePS is the latest member of the SNePS Family of KR systems.

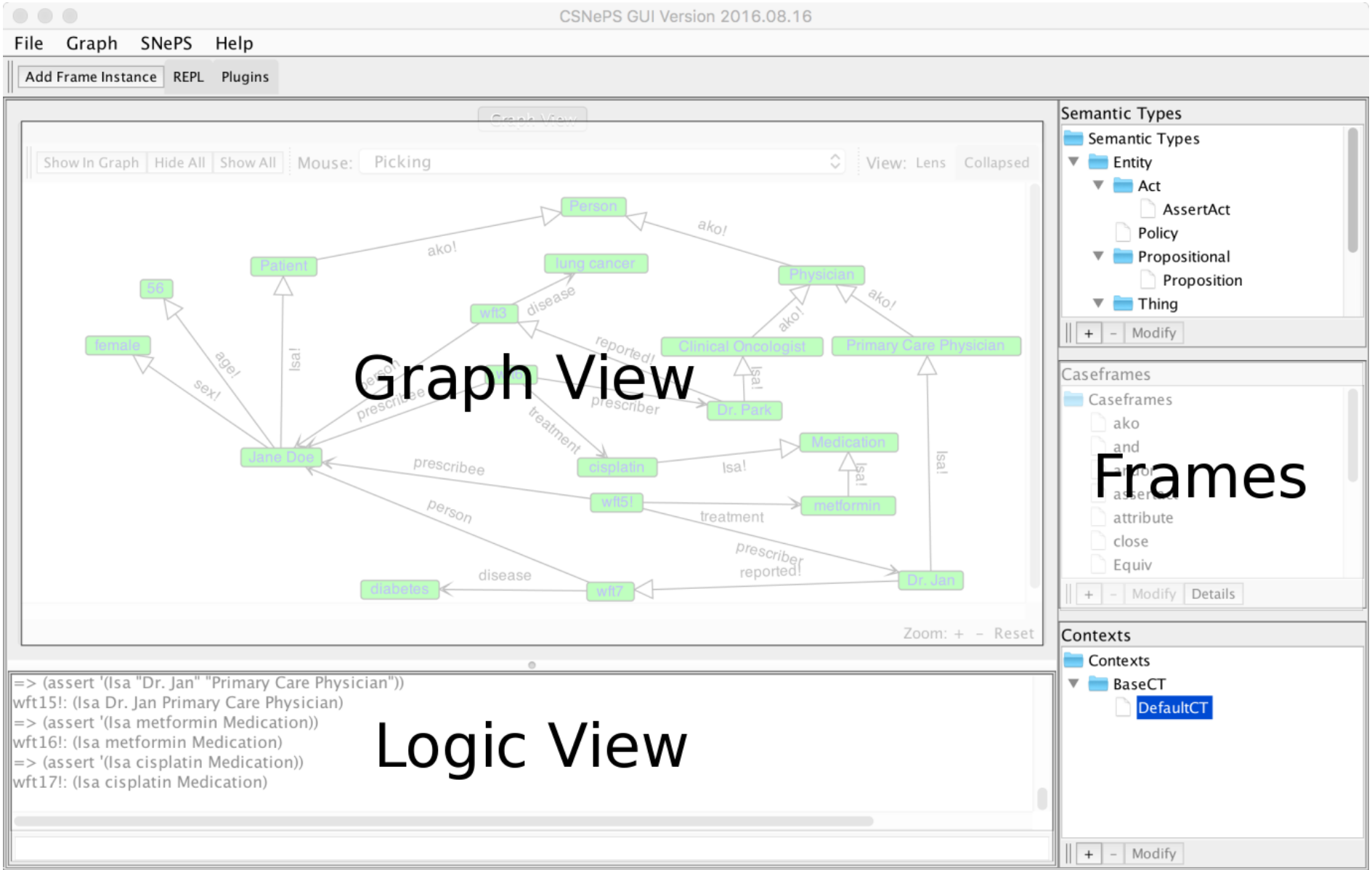
Implemented by Dan Schlegel.

The (C)SNePS KB can be thought of as simultaneously being:

- Term Logic based,
- Assertional Frame based,
- Propositional Graph based.

We have created a user interface which uses all three:

CSNePS GUI


CSNePS GUI Version 2016.08.16
File
Graph
SNePS
Help
Add Frame Instance
REPL
Plugins
Show In Graph
Hide All
Show All
Mouse:
Picking
View:
Lens
Collapsed
Person
Patient
lung cancer
Physician
56
wft3
disease
female
Clinical Oncologist
Primary Care Physician
age!
sex!
isa!
reported!
prescribee
prescriber
treatment
Dr. Park
Medication
Jane Doe
cisplatin
wft5!
metformin
person
Dr. Jan
diabetes
wft7
Graph View
Zoom: + - Reset
=> (assert '(Isa "Dr. Jan" "Primary Care Physician"))
wft15!: (Isa Dr. Jan Primary Care Physician)
=> (assert '(Isa metformin Medication))
wft16!: (Isa metformin Medication)
=> (assert '(Isa cisplatin Medication))
wft17!: (Isa cisplatin Medication)
Logic View
Semantic Types
Entity
Act
AssertAct
Policy
Propositional
Proposition
Thing
Modify
Caseframes
ako
and
attribute
close
Equiv
Details
Frames
Contexts
BaseCT
DefaultCT

Outline

KB as set of Logical Expressions

The CSNePS KB is a set of logical expressions:

- Atomic terms
  - Individual constants denoting entities in domain including classes, propositions, and some relations
- Arbitrary and indefinite terms [Shapiro, KR2004]
- Functional terms including
  - terms denoting individuals
  - 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 CSNePS KB is a set of logical expressions:

- Atomic terms
  - Individual constants denoting entities in domain including classes, propositions, and some relations
- Arbitrary and indefinite terms [Shapiro, KR2004]
- Functional terms including
  - terms denoting individuals
  - 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 CSNePS KB is a set of logical expressions:

- Atomic terms
  - Individual constants denoting entities in domain including classes, propositions, and some relations
- Arbitrary and indefinite terms [Shapiro, KR2004]
- Functional terms
  including
  - terms denoting individuals
  - 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 CSNePS KB is a set of logical expressions:

- Atomic terms
  - Individual constants denoting entities in domain including classes, propositions, and some relations
- Arbitrary and indefinite terms [Shapiro, KR2004]
- Functional terms
  including
  - terms denoting individuals
  - 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 CSNePS KB is a set of logical expressions:

- Atomic terms
  - Individual constants denoting entities in domain including classes, propositions, and some relations
- Arbitrary and indefinite terms [Shapiro, KR2004]
- Functional terms
  including
  - terms denoting individuals
  - 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 CSNePS KB is a set of logical expressions:

- Atomic terms
  - Individual constants denoting entities in domain including classes, propositions, and some relations
- Arbitrary and indefinite terms [Shapiro, KR2004]
- Functional terms
  including
  - terms denoting individuals
  - 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”.

Example Input

```
> (assert '(Isa "Jane Doe" Patient))
wft13!: (Isa Jane Doe Patient)

> (assert '(Isa "Dr. Park" "Clinical Oncologist"))
wft16!: (Isa Dr. Park Clinical Oncologist)

> (assert '(Isa "lung cancer" disease))
wft25!: (Isa lung cancer disease)

> (assert '(reported "Dr. Park"
                (hasDisease "Jane Doe" "lung cancer")))
wft27!: (reported Dr. Park
            (hasDisease Jane Doe lung cancer))
```

"Dr. Park, a clinical oncologist, reported that Jane Doe, a patient, has lung cancer, a disease."

Note: `wft27` gives meta-information.

Outline

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.
  - Dr. Jan prescribed metformin to Jane Doe.
  - Metformin was prescribed to Jane Doe by Dr. Jan.
  - Jane Doe has a prescription of metformin from Dr. Jan.
- 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.
  - Dr. Jan prescribed metformin to Jane Doe.
  - Metformin was prescribed to Jane Doe by Dr. Jan.
  - Jane Doe has a prescription of metformin from Dr. Jan.
- 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.
  - Dr. Jan prescribed metformin to Jane Doe.
  - Metformin was prescribed to Jane Doe by Dr. Jan.
  - Jane Doe has a prescription of metformin from Dr. Jan.
- 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.
  - Dr. Jan prescribed metformin to Jane Doe.
  - Metformin was prescribed to Jane Doe by Dr. Jan.
  - Jane Doe has a prescription of metformin from Dr. Jan.
- 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.
  - Dr. Jan prescribed metformin to Jane Doe.
  - Metformin was prescribed to Jane Doe by Dr. Jan.
  - Jane Doe has a prescription of metformin from Dr. Jan.
- We will use “caseframe” for their “frame”
- and use “frame” for an instantiated caseframe.

Components of Caseframes

Definition

A caseframe has

- A name[a]
- A semantic type (sort)
  The type of the instances of the caseframe
- An ordered list of named slots

[a] Temporary simplification for GUI.

Examples of Caseframes

Example

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

Example

`prescription` is a caseframe of type `Proposition`
with slots `prescribee`, `treatment`, and `prescriber`.

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 '(prescription
                "Jane Doe" metformin "Dr. Jan"))
```

creates an instance of the `prescription` caseframe

whose `prescribee` slot contains the filler `Jane Doe`,

whose `treatment` slot contains the filler `metformin`,

and whose `prescriber` slot contains the filler `Dr. Jan`.

Outline

Propositional Graphs

A way of visualizing and traversing the frames/logical expressions.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Arbitrary and Indefinite terms
  - 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.
- Collapse propositional node to binary edge labeled with relation where possible.

Propositional Graphs

A way of visualizing and traversing the frames/logical expressions.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Arbitrary and Indefinite terms
  - 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.
- Collapse propositional node to binary edge labeled with relation where possible.

Propositional Graphs

A way of visualizing and traversing the frames/logical expressions.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Arbitrary and Indefinite terms
  - 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.
- Collapse propositional node to binary edge labeled with relation where possible.

Propositional Graphs

A way of visualizing and traversing the frames/logical expressions.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Arbitrary and Indefinite terms
  - 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.
- Collapse propositional node to binary edge labeled with relation where possible.

Propositional Graphs

A way of visualizing and traversing the frames/logical expressions.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Arbitrary and Indefinite terms
  - 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.
- Collapse propositional node to binary edge labeled with relation where possible.

Propositional Graphs

A way of visualizing and traversing the frames/logical expressions.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Arbitrary and Indefinite terms
  - 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.
- Collapse propositional node to binary edge labeled with relation where possible.

Propositional Graphs

A way of visualizing and traversing the frames/logical expressions.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Arbitrary and Indefinite terms
  - 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.
- Collapse propositional node to binary edge labeled with relation where possible.

Propositional Graphs

A way of visualizing and traversing the frames/logical expressions.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Arbitrary and Indefinite terms
  - 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.
- Collapse propositional node to binary edge labeled with relation where possible.

Propositional Graphs

A way of visualizing and traversing the frames/logical expressions.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Arbitrary and Indefinite terms
  - 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.
- Collapse propositional node to binary edge labeled with relation where possible.

Propositional Graphs

A way of visualizing and traversing the frames/logical expressions.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Arbitrary and Indefinite terms
  - 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.
- Collapse propositional node to binary edge labeled with relation where possible.

Propositional Graphs

A way of visualizing and traversing the frames/logical expressions.

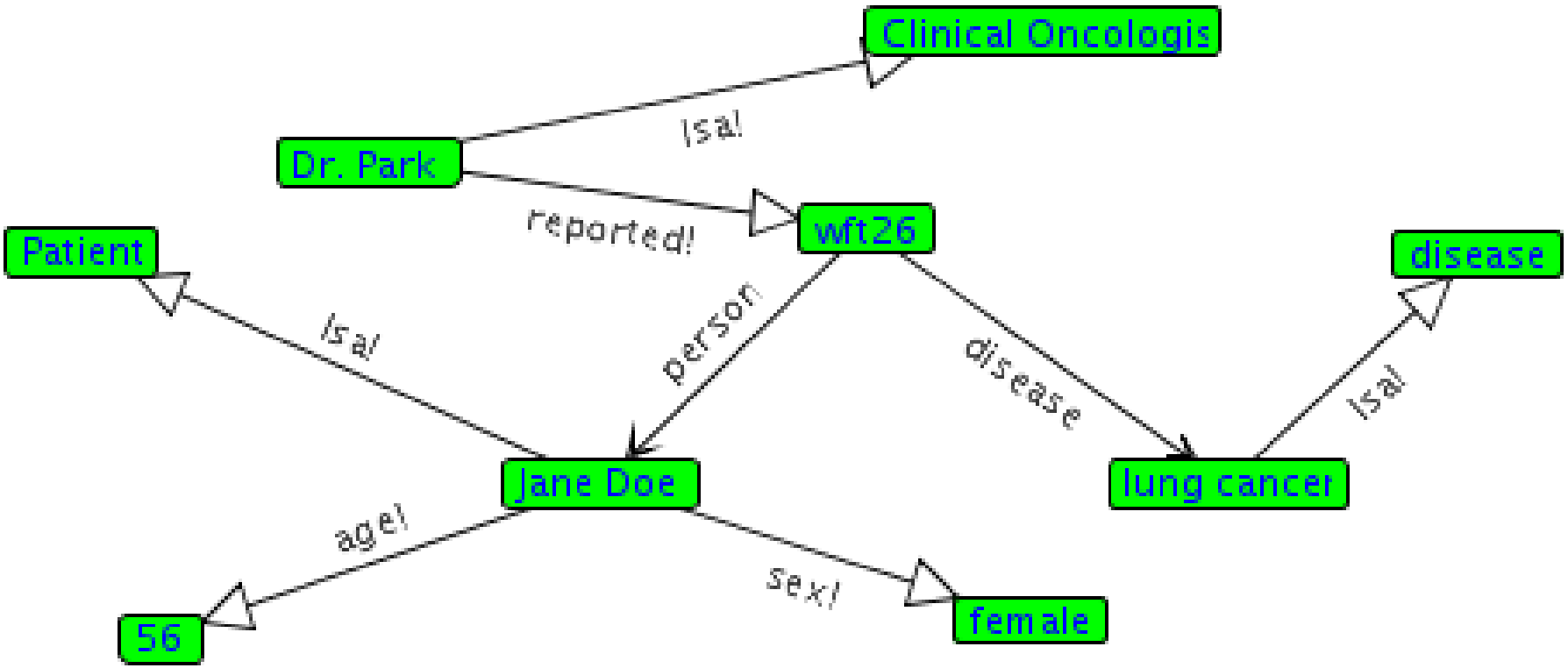
- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Arbitrary and Indefinite terms
  - Functional terms (frames)
  - Proposition-denoting functional terms
- Node ID is
  - symbol
  - frame name (`wft`*i* [!])
- Edges drawn
  from the node corresponding to the frame,
  to the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

Propositional Graphs

A way of visualizing and traversing the frames/logical expressions.

- Directed Acyclic Graph
- Every term is a node.
  - Individual constants
  - Arbitrary and Indefinite terms
  - 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.
- Collapse propositional node to binary edge labeled with relation where possible.

Example Propositional Graph



"Dr. Park, a clinical oncologist, reported that Jane Doe, a 56-year old female patient, has lung cancer, a disease."

Outline

N-ary Relations

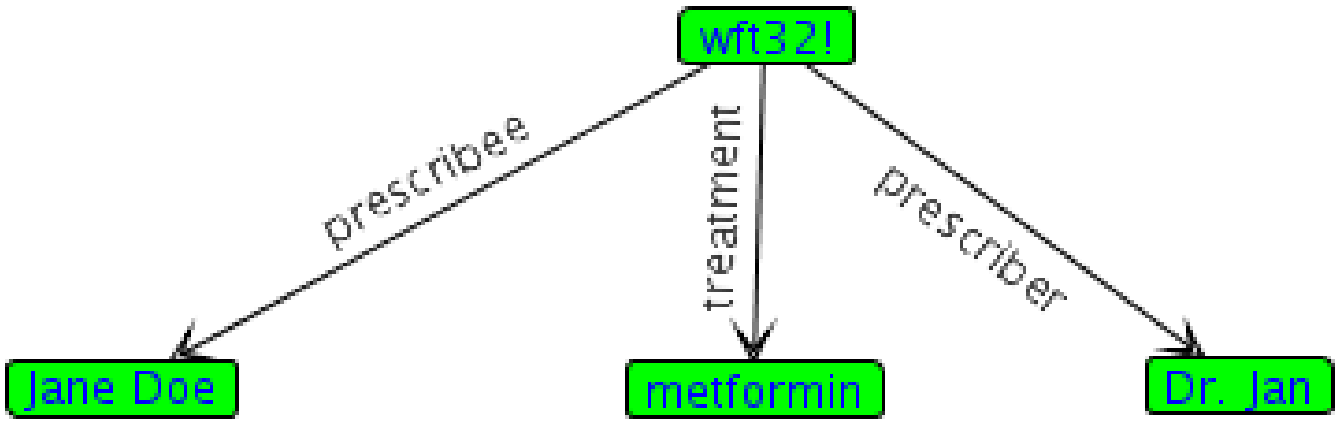
A relation among *n* arguments

```
(prescription "Jane Doe" metformin "Dr. Jan")
```

corresponds to a frame with *n* slots

```
wft31
    prescribee:  Jane Doe
    treatment:  metformin
    prescriber:  Dr. Jan
```

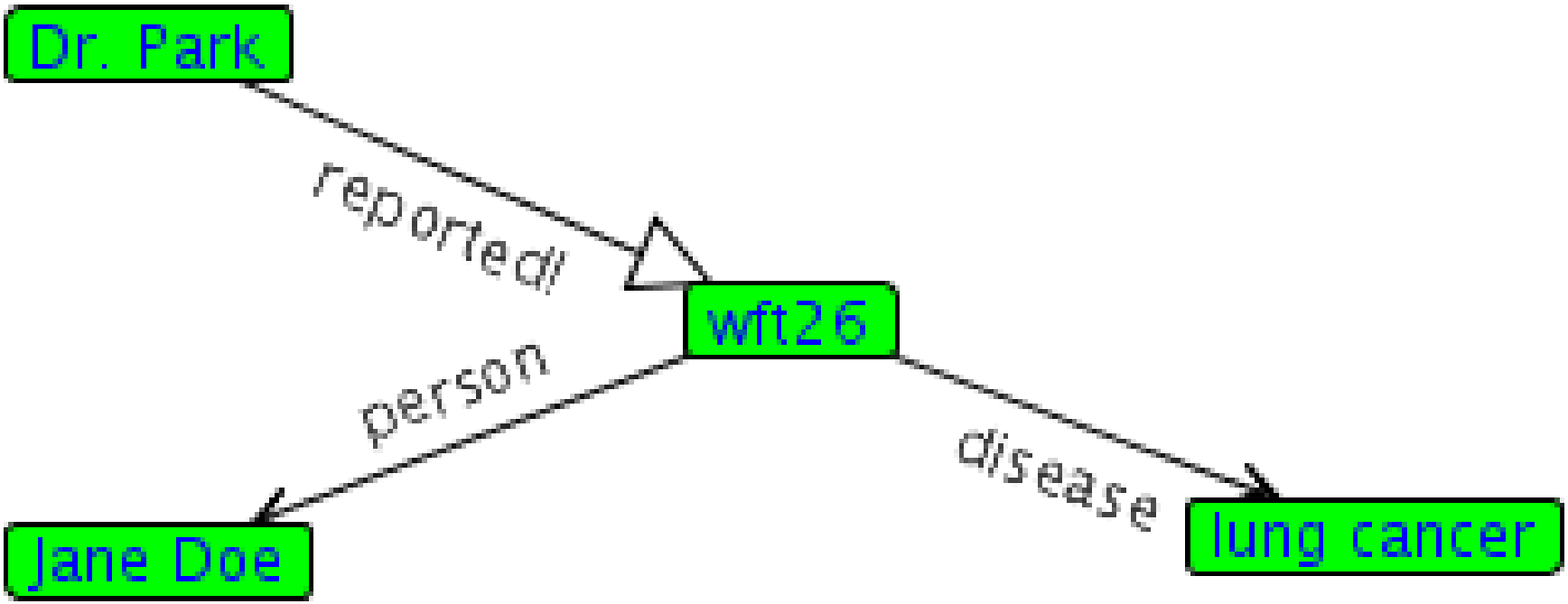
corresponds to a node with *n* outgoing arcs.



Representing Pedigree

The pedigree of a proposition *P*
is represented as an assertion about *P*.

Example: Dr. Park reported that Jane Doe has lung cancer.



Negation

Issues:

- What is the relation between a proposition and its negation?
  - For some schemes, there is none.

```
(HasDisease "Jane Doe" arthritis)
(DoesntHaveDisease "Jane Doe" arthritis)

(shouldTake "Jane Doe" metformin)
(shouldNotTake "Jane Doe" metformin)
```

- Negated arcs in hierarchies present reasoning difficulties.

Negation

Issues:

- What is the relation between a proposition and its negation?
  - For some schemes, there is none.

```
(HasDisease "Jane Doe" arthritis)
(DoesntHaveDisease "Jane Doe" arthritis)

(shouldTake "Jane Doe" metformin)
(shouldNotTake "Jane Doe" metformin)
```

- Negated arcs in hierarchies present reasoning difficulties.

Negation

Issues:

- What is the relation between a proposition and its negation?
  - For some schemes, there is none.

```
(HasDisease "Jane Doe" arthritis)
(DoesntHaveDisease "Jane Doe" arthritis)

(shouldTake "Jane Doe" metformin)
(shouldNotTake "Jane Doe" metformin)
```

- Negated arcs in hierarchies present reasoning difficulties.

Negation

Issues:

- What is the relation between a proposition and its negation?
  - For some schemes, there is none.

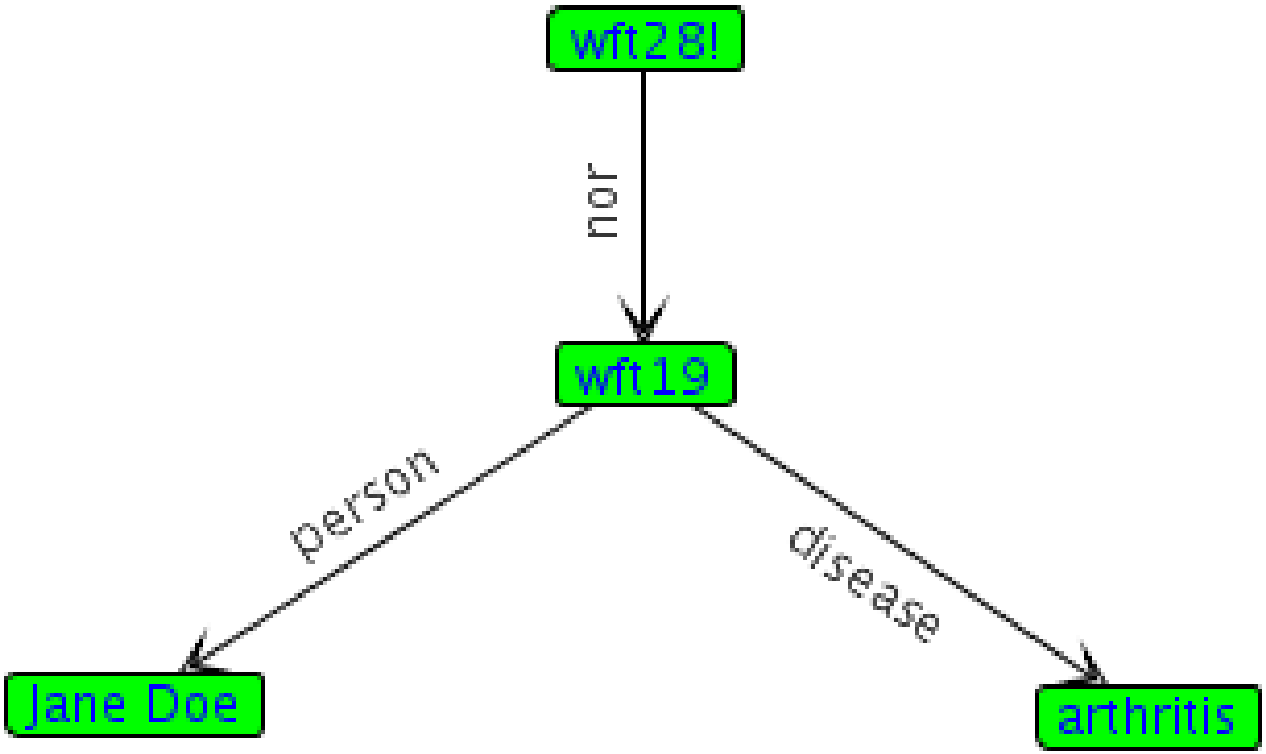
```
(HasDisease "Jane Doe" arthritis)
(DoesntHaveDisease "Jane Doe" arthritis)

(shouldTake "Jane Doe" metformin)
(shouldNotTake "Jane Doe" metformin)
```

- Negated arcs in hierarchies present reasoning difficulties.

SNePS Approach to Negation

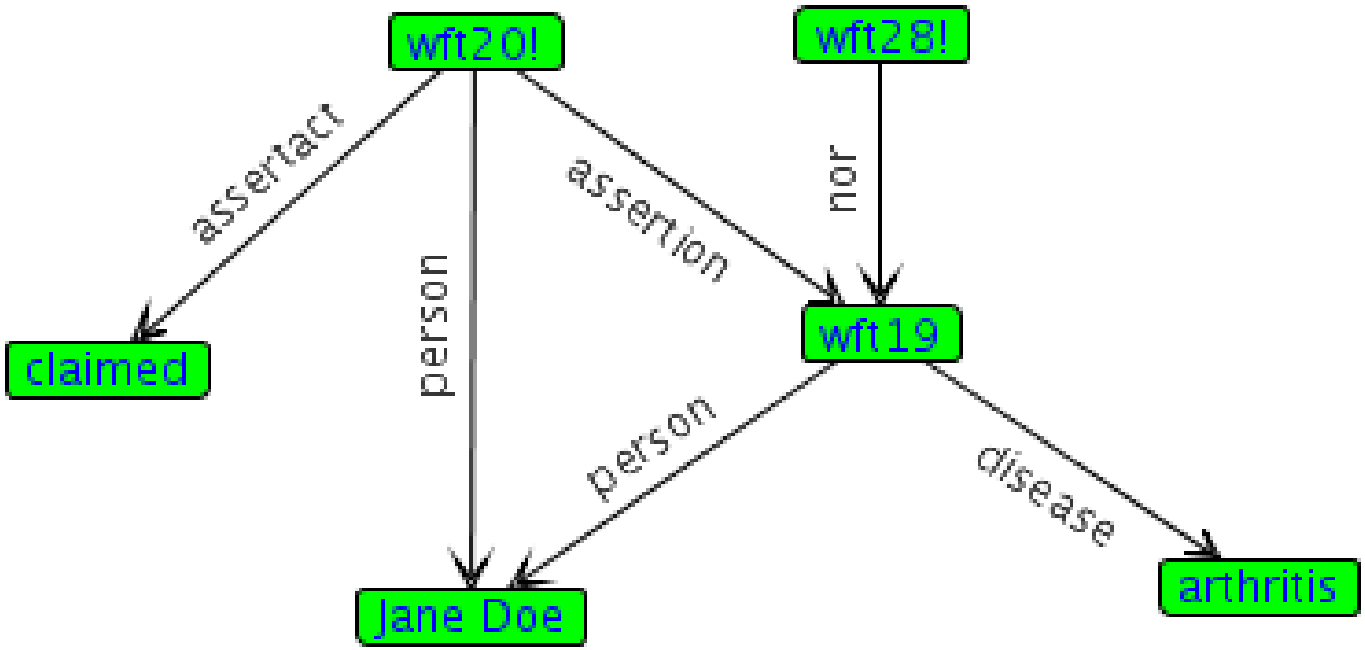
`not` is a proposition-valued functional term
that takes a proposition as argument.

```
(not (HasDisease "Jane Doe" arthritis))
```



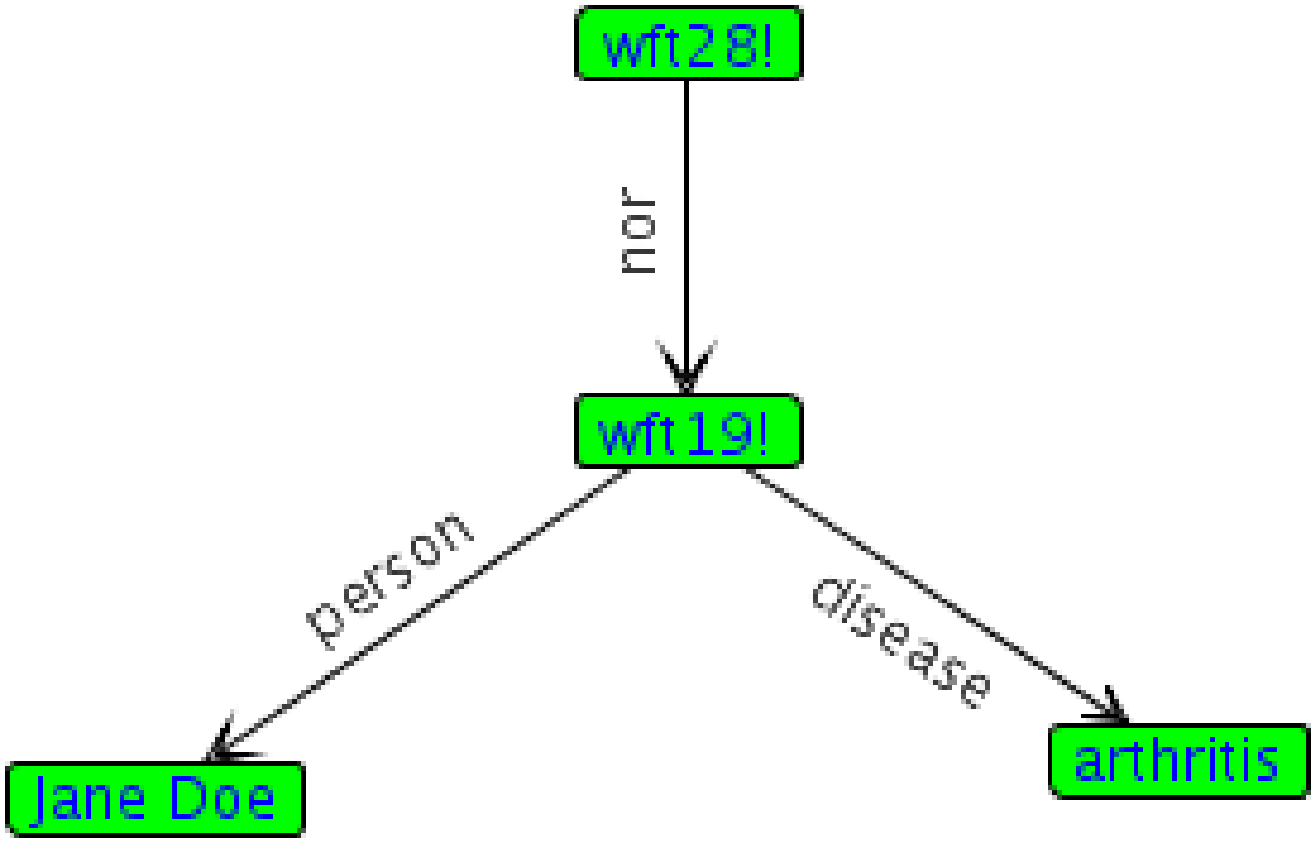
Benefits of SNePS Approach 1

Can represent proposition and its negation independently.
Example: Jane Doe claimed that she has arthritis, but she doesn't.



Benefits of SNePS Approach 2

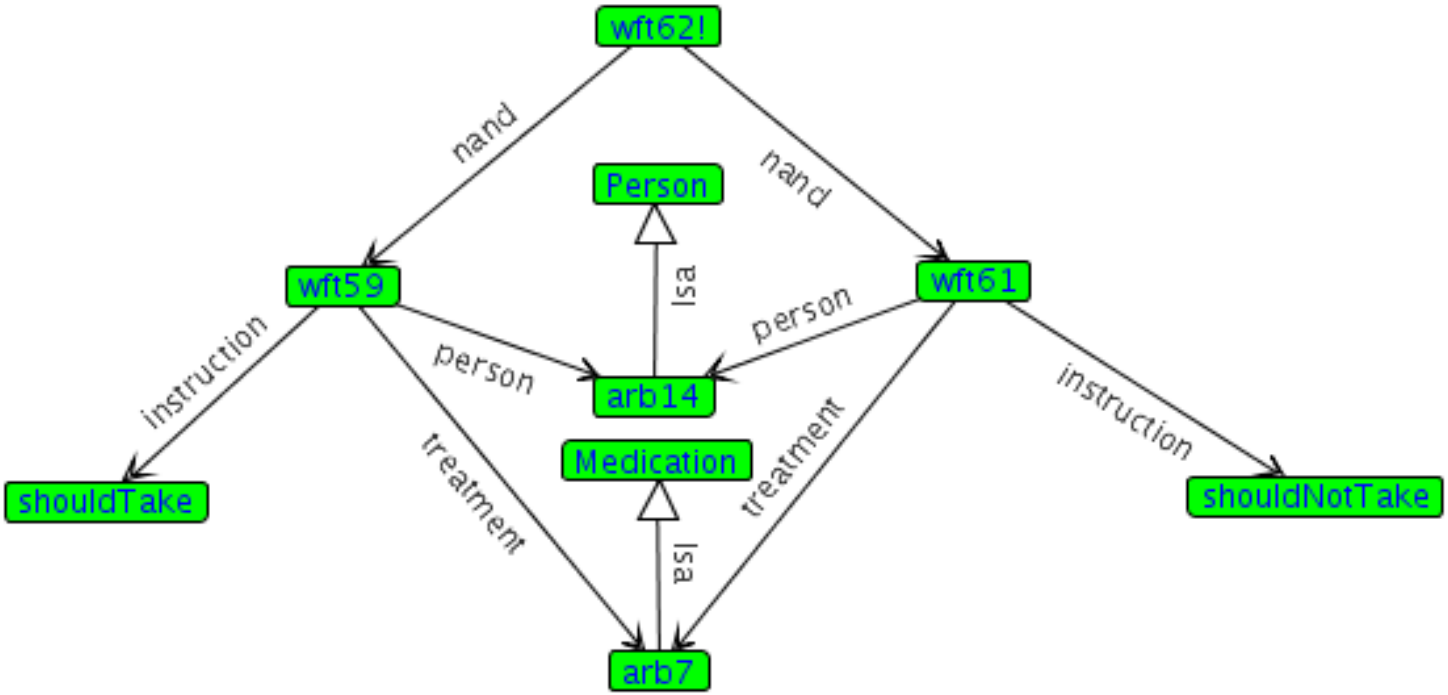
Can recognize contradictions.
Example: Jane Doe has arthritis, and she doesn't.



Representing Rules

A rule (non-atomic proposition)
is a proposition with a logical operator as relation
and other propositions as arguments.

```
(assert '(nand (shouldTake (any p Person) (any m Medication))
               (shouldNotTake p m)))
```



Outline

Conclusions

When choosing (or starting to use) a KR system,

- Consider
  - what the system invites you to focus on:
    - objects,
    - classes,
    - propositions;
  - how the system presents information:
    - logical expressions,
    - frames,
    - graphs;
  - how it will deal with:
    - n-ary relations,
    - negation,
    - pedigree,
    - attributes,
    - other meta-information.
- Don't be fooled by superficial differences.

Conclusions

When choosing (or starting to use) a KR system,

- Consider
  - what the system invites you to focus on:
    - objects,
    - classes,
    - propositions;
  - how the system presents information:
    - logical expressions,
    - frames,
    - graphs;
  - how it will deal with:
    - n-ary relations,
    - negation,
    - pedigree,
    - attributes,
    - other meta-information.
- Don't be fooled by superficial differences.

Conclusions

When choosing (or starting to use) a KR system,

- Consider
  - what the system invites you to focus on:
    - objects,
    - classes,
    - propositions;
  - how the system presents information:
    - logical expressions,
    - frames,
    - graphs;
  - how it will deal with:
    - n-ary relations,
    - negation,
    - pedigree,
    - attributes,
    - other meta-information.
- Don't be fooled by superficial differences.

Conclusions

When choosing (or starting to use) a KR system,

- Consider
  - what the system invites you to focus on:
    - objects,
    - classes,
    - propositions;
  - how the system presents information:
    - logical expressions,
    - frames,
    - graphs;
  - how it will deal with:
    - n-ary relations,
    - negation,
    - pedigree,
    - attributes,
    - other meta-information.
- Don’t be fooled by superficial differences.

For More Information

Shapiro: `http://www.cse.buffalo.edu/~shapiro/`
Schlegel: `http://danielschlegel.org/`
SNePS Research Group: `http://www.cse.buffalo.edu/sneps/`