Constructions and Metaphor: Integrating MetaNet and Embodied Construction Grammar

Ellen Dodge
Oana David
Elise Stickles
Eve Sweetser
Overview

Background
• MetaNet
• Embodied Construction Grammar (ECG)

Analysis of examples:
• Literal example:
  *The cat attacked the mouse*

• Metaphor examples:
  *Poverty attacked children*
  *The government attacked poverty.*
MetaNet

- Conceptual metaphors are typically described in the format of TARGET IS SOURCE.

- Examples:
  - POVERTY IS A BEAST
  - POVERTY IS A DISEASE
  - SOCIAL PROBLEMS ARE HARMFUL AGENTS
  - SOCIAL PROBLEMS ARE PHYSICAL HARM
MetaNet

More general

- SOCIAL PROBLEMS ARE PHYSICAL HARM
  - SOCIAL PROBLEMS ARE DISEASES
    - ADDRESSING SOCIAL PROBLEMS IS TREATING A DISEASE
  - SOCIAL PROBLEMS ARE HARMFUL AGENTS
    - SOCIAL PROBLEMS ARE BEASTS

More specific
MetaNet

Metaphor: SOCIAL PROBLEMS ARE DISEASES

Social Problems

- social problem
- social group
- effect_of_problem

Disease

- disease
- diseased_entity
- effect_of_disease
Embodied Construction Grammar (ECG)

Meaning is central:
- Constructional meaning is represented using schemas

- Formalizes Cognitive Linguistics insights, e.g.:
  - image schemas (Johnson, Lakoff)
  - frame semantics (Fillmore)
  - profiling, perspective (Talmy, Langacker)

- ECG meaning representations are compatible with MetaNet schemas and metaphors, by design
Embodied Construction Grammar (ECG)

**Unification-based construction grammar:**
- Computationally-implemented, construction-based analysis of sentence meaning
- Analysis of a given sentence takes constructional constraints on **form and meaning** into account to determine the ‘best-fitting’ set of constructions
- Constructional unification results in a Semantic Specification of sentence meaning.
- Semantic Specification is in the form of schemas, roles, bindings, values, and role-filler information
Analysis of literal example: The cat attacked the mouse

Example instantiates several constructions:
- Noun, Determiner, NP
- **Verb**
- **TransitiveCausation**
- Declarative

Constructional meaning is represented using schemas:
- Verb meaning
  - “Attack” ← Attacking schema
- Argument structure cxn meaning is more general:
  TransitiveCausation ← Cause-effect action schema
Analysis of literal example:

The cat attacked the mouse

- **Attacker**
  - Role type: animate entity

- **Attackee**
  - Role type: physical entity

- **Attacking_action**
Analysis of literal example:

*The cat attacked the mouse*
Analysis of literal example:
The cat attacked the mouse
Analysis of literal example:

*The cat attacked the mouse*

**CONSTRUCTION** ActiveTrans_Causation_central

**subcase of** Active_Transitive

**CONSTRUCTIONAL** constituents

verb: Verb
np: NP

**FORM** constraints

verb **before** np

**MEANING:** Cause-effect_action

constraints

cxn_whole <!--> verb
agent <!--> do_np
affectee <!--> profiled_Participant
Analysis of literal example:

*The cat attacked the mouse*
Analysis of literal example: The cat attacked the mouse
Analysis of literal example: 

*The cat attacked the mouse*

Unification of instantiated constructions:

- A-S cxn specifies bindings to Cause-effect_action roles:
  
  \[
  \text{profiledParticipant} = \text{agent} \\
  \text{np} = \text{affectee}
  \]

- Unification with verb ‘extends’ these bindings to the attacking schema roles
  
  \[
  \text{profiledParticipant} = \text{agent} = \text{attacker} \\
  \text{np} = \text{affectee} = \text{attackee}
  \]

- Fillers of these roles need to meet role type constraints:
  - subject needs to type ‘animate entity’
  - direct object np needs to be of type ‘physical entity’
Analysis of literal example:
The cat attacked the mouse
Metaphor example: *Poverty attacked children.*

- “Poverty” cxn has role type: poverty

- “Poverty” cxn will not unify with the previous A-S cxn since it conflicts with the type constraints of the ‘agent’ and ‘attacker’ roles (i.e. it is not an animate entity).
Metaphor example: *Poverty attacked children.*
Analysis of literal example: *Poverty attacked children*

**SCHEMA**  Metaphor_SocialProblemsAreHarmfulAgents

**subcase of**  Metaphor_AbstractHarmerIsPhysicalHarmer

**roles**

source: PhysicalHarm

target: SocialProblems

**constraints**

mapping1.source <-> harmAgent

mapping1.target <-> socialProblem

mapping2.source <-> harmed

mapping2.target <-> socialGroup
Metaphor example: 
Poverty attacked children.

**CONSTRUCTION** Met_Trans_SocialProblemsAreHarmfulAgents

*subcase of* Metaphorical_ActiveTransitiveCausation

**MEANING**

*evokes* met_SocialProblemsAreHarmfulAgent

*constraints*

met.target <---> self
met.source <---> verb
met.target.socialGroup <---> np
met.target.socialProblem <---> profiledParticipant
Metaphor example: Poverty attacked children.
Metaphor example: Poverty attacked children.
Metaphor example: 
*Poverty attacked children.*

- **ABSTRACT HARMERS ARE PHYSICAL HARMERS**
- **SOCIAL PROBLEMS ARE HARMFUL AGENTS**
- **SOCIAL PROBLEMS ARE ATTACKERS**
Metaphor example: poverty attacks

When poverty attacks, it tears at the very fabric of a community.
Metaphor example:
To end inequality, **attack poverty**, not the 1 percent.
Metaphor example: attack poverty

**SCHEMA** Metaphor<AddressingSocialProblemsIsAttacking

**subcase of** Metaphor_AbstractHarmeeIsPhysicalHarmee

**roles**
- source: Attacking
- target: AddressingSocialProblems

**constraints**
- mapping1.source <--> attacker
- mapping1.target <--> problemSolver

- mapping2.source <--> attackee
- mapping2.target <--> socialProblem
Metaphor example:  
*attack poverty*

**CONSTRUCTION** Met_Trans_AddressingSocialProblemsIsAttacking  
*subcase of* Metaphor_AbstractHarmIsPhysicalHarm_affected

**MEANING**

*evokes* Metaphor_AddressingSocialProblemsIsAttacking  
as  
*met*

*constraints*

met.source  *<--->* verb  
met.target  *<--->* self  
met.target.socialProblem  *<--->* np  
met.target.problemSolver  *<--->* profiledParticipant
Metaphor example: *attack poverty*

To end inequality, **attack poverty**, not the 1 percent.
Concluding remarks

ECG and MetaNet are complementary resources:
- Both analyze and represent conceptual structure in compatible ways, by design
- MetaNet provides rich, structured networks of schemas and metaphors
- ECG supports construction-based, deep semantic analysis of literal and metaphorical sentences

Schema and metaphor networks that capture different levels of generalization enable us to define:
- general A-S constructions that capture broad generalizations
- subcases that more closely reflect actual patterns of metaphor expression
Acknowledgments

The MetaNet Analysis team: George Lakoff, Jisup Hong, Karie Moorman

ECG team at ICSI: Jerome Feldman, Luca Gilardi

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12-C-0022.
The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.
Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

Supported by the Office Naval Research (ONR) N000141110416
Selected References


schema Metaphor_SocialProblemsAreHarmfulAgents
  subcase of Metaphor_AbstractCauseHarmIsPhysicalCauseHarm_causalAgent
  roles
    source: PhysicalHarm@SCHEMA
    target: SocialProblem@SCHEMA
  name //inherited from Metaphor_pairTest
  mapping1: Mapping@SCHEMA //inherited from Metaphor_pairTest
  mapping2: Mapping@SCHEMA //inherited from Metaphor_pairTest
  mapping3: Mapping@SCHEMA //inherited from Metaphor_pairTest
  mapping4: Mapping@SCHEMA //inherited from Metaphor_pairTest
  constraints
    target.protagonist.ontological-category <-- @socialProblem
    name <-- "Social problems are Harmful agents"
    mapping1.source <-- source.protagonist //inherited from
      Metaphor_AbstractCauseHarmIsPhysicalCauseHarm_causalAgent
    mapping1.target <-- target.protagonist //inherited
    mapping2.source <-- source.protagonist2 //inherited
    mapping2.target <-- target.protagonist2 //inherited
    target.protagonist.ontological-category <-- @abstract  //inherited
CONSTRUCTION Metaphorical_Transitive_SocialProblemsAreHarmfulAgents
subcase of Metaphorical_ActiveTransitiveCausation2
CONSTRUCTIONAL: VerbFeatures@SCHEMA
  constituents
  np: NP@CONSTRUCTION [ [ 1.0 .9 ] ]  //inherited from ActiveTransitive
  v: Verb@CONSTRUCTION  //inherited from ArgumentStructure
  constraints
  v.voice <-- NotPassive  //inherited from ActiveTransitive
  np.features.case <-- "obj"  //inherited from ActiveTransitive
  self.features <-- v.features  //inherited from ArgumentStructure
  self.verbform <-- v.verbform  //inherited from ArgumentStructure
  self.voice <-- v.voice  //inherited from ArgumentStructure
FORM: UNTYPED
  constraints
  v.f before np.f  //inherited from ActiveTransitive
MEANING: Process@SCHEMA
  evokes Metaphor_SocialProblemsAreHarmfulAgents@SCHEMA as met
  evokes A123@SCHEMA as s  //inherited from ActiveTransitive
  evokes EventDescriptor@SCHEMA as ed  //inherited from VP
  constraints
  v.m <-- self.met.source  //inherited from Metaphorical_ActiveTransitiveCausation2
  self.m <-- self.met.target  //inherited from Metaphorical_ActiveTransitiveCausation2
  s.a1 <-- self.met.target.protagonist  //inherited from Metaphorical_ActiveTransitiveCausation2
  s.a2 <-- self.met.target.protagonist2  //inherited from Metaphorical_ActiveTransitiveCausation2
  s.a1 <-- ed.profiledParticipant  //inherited from ActiveTransitive
  s.a2 <-- np.m  //inherited from ActiveTransitive
  v.m.p-features.voice <-- @active  //inherited from ActiveTransitive
  s.a2.ontological-category <-- np.m.ontological-category  //inherited from ActiveTransitive
  v.m <-- ed.profiledProcess  //inherited from ArgumentStructure
  self.m.p-features <-- v.m.p-features  //inherited from ArgumentStructure
  self.m <-- ed.eventProcess  //inherited from VP
  self.m.ed <-- ed  //inherited from VP
CONSTRUCTION Metaphorical_Transitive_AbstractCauseHarmIsPhysicalCauseHarm_affected
subcase of ActiveTransitive
CONSTRUCTIONAL: VerbFeatures@SCHEMA
   constituents
      np: NP@CONSTRUCTION [ [ 1.0 .9 ] ]  //inherited from ActiveTransitive
      v: Verb@CONSTRUCTION  //inherited from ArgumentStructure
   constraints
      v.voice <-- NotPassive  //inherited from ActiveTransitive
      np.features.case <-- "obj"  //inherited from ActiveTransitive
      self.features <-- v.features  //inherited from ArgumentStructure
      self.verbform <-- v.verbform  //inherited from ArgumentStructure
      self.voice <-- v.voice  //inherited from ArgumentStructure

FORM: UNTYPED
   constraints
      v.f before np.f  //inherited from ActiveTransitive

MEANING: Process@SCHEMA
   evokes Metaphor_AbstractCauseHarmIsPhysicalCauseHarm_affected@SCHEMA as met
   evokes A123@SCHEMA as s  //inherited from ActiveTransitive
   evokes EventDescriptor@SCHEMA as ed  //inherited from VP
   constraints
      v.m <-- self.met.source
      self.m <-- self.met.target
      s.a1 <-- self.met.target.protagonist
      s.a2 <-- self.met.target.protagonist2
      s.a2.ontological-category <-- self.met.target.protagonist2.ontological-category
      s.a1 <-- ed.profiledParticipant  //inherited from ActiveTransitive
      s.a2 <-- np.m  //inherited from ActiveTransitive
      v.m.p-features.voice <-- @active  //inherited from ActiveTransitive
      s.a2.ontological-category <-- np.m.ontological-category  //inherited from ActiveTransitive
      v.m <-- ed.profiledProcess  //inherited from ArgumentStructure
      self.m.p-features <-- v.m.p-features  //inherited from ArgumentStructure
      self.m <-- ed.eventProcess  //inherited from VP
      self.m.ed <-- ed  //inherited from VP