Model Theory of HPSG grammars

Johannes Dellert

Seminar für Sprachwissenschaft, Universität Tübingen

July 16, 2007

What are those feature structures?

- we have been dealing wih the notion of feature structures quite naturally

What are those feature structures?

- we have been dealing wih the notion of feature structures quite naturally
- but what is it that they represent?

What are those feature structures?

- we have been dealing wih the notion of feature structures quite naturally
- but what is it that they represent?
- in how far are we talking about language when building HPSG grammars?

What are those feature structures?

- we have been dealing wih the notion of feature structures quite naturally
- but what is it that they represent?
- in how far are we talking about language when building HPSG grammars?
- what are the conclusions to draw from possible answers?

Outline

- comparison of the formal foundations of HPSG 87 and 94

Outline

- comparison of the formal foundations of HPSG 87 and 94
    - quick glimpse at the ideas behind formalizing HPSG 87

Outline

- comparison of the formal foundations of HPSG 87 and 94
    - quick glimpse at the ideas behind formalizing HPSG 87
- closer look at the different concepts for formalizing the meaning of HPSG 94

Outline

- comparison of the formal foundations of HPSG 87 and 94
  - quick glimpse at the ideas behind formalizing HPSG 87
- closer look at the different concepts for formalizing the meaning of HPSG 94
  - Basic mechanism: SRL

Outline

- comparison of the formal foundations of HPSG 87 and 94
  - quick glimpse at the ideas behind formalizing HPSG 87
- closer look at the different concepts for formalizing the meaning of HPSG 94
  - Basic mechanism: SRL
  - King 1999: Exhaustive models

Outline

- comparison of the formal foundations of HPSG 87 and 94
  - quick glimpse at the ideas behind formalizing HPSG 87
- closer look at the different concepts for formalizing the meaning of HPSG 94
  - Basic mechanism: SRL
  - King 1999: Exhaustive models
  - Pollard & Sag 1994: Feature structures as object types

Outline

- comparison of the formal foundations of HPSG 87 and 94
  - quick glimpse at the ideas behind formalizing HPSG 87
- closer look at the different concepts for formalizing the meaning of HPSG 94
  - Basic mechanism: SRL
  - King 1999: Exhaustive models
  - Pollard & Sag 1994: Feature structures as object types
  - Pollard 1999: Strong Generative Capacity

Outline

- comparison of the formal foundations of HPSG 87 and 94
  - quick glimpse at the ideas behind formalizing HPSG 87
- closer look at the different concepts for formalizing the meaning of HPSG 94
  - Basic mechanism: SRL
  - King 1999: Exhaustive models
  - Pollard & Sag 1994: Feature structures as object types
  - Pollard 1999: Strong Generative Capacity
- hints on how the different views can be dealt with

HPSG as in P & S 1987

- feature structures represent partial information on linguistic objects

HPSG as in P & S 1987

- feature structures represent partial information on linguistic objects
- task of the grammar: specify the **knowledge** of a mature speaker of a language

HPSG as in P & S 1987

- feature structures represent partial information on linguistic objects
- task of the grammar: specify the **knowledge** of a mature speaker of a language
- no commitment about the nature of the entities that constitute a natural language

HPSG as in P & S 1987

- feature structures represent partial information on linguistic objects
- task of the grammar: specify the **knowledge** of a mature speaker of a language
- no commitment about the nature of the entities that constitute a natural language
- HPSG 87 operates only on **information about** linguistic objects

HPSG as in P & S 1987

- feature structures represent partial information on linguistic objects
- task of the grammar: specify the **knowledge** of a mature speaker of a language
- no commitment about the nature of the entities that constitute a natural language
- HPSG 87 operates only on **information about** linguistic objects

HPSG as in P & S 1994

- structures represent objects that constitute the language of the grammar

HPSG as in P & S 1994

- structures represent objects that constitute the language of the grammar
- task of the grammar: specifying the **object types** of a natural language

HPSG as in P & S 1994

- structures represent objects that constitute the language of the grammar
- task of the grammar: specifying the **object types** of a natural language
- feature structures are idealizations of equivalence classes of well-formed linguistic entities

HPSG as in P & S 1994

- structures represent objects that constitute the language of the grammar
- task of the grammar: specifying the **object types** of a natural language
- feature structures are idealizations of equivalence classes of well-formed linguistic entities
- feature structures are not partial, but **complete representations** of linguistic entities

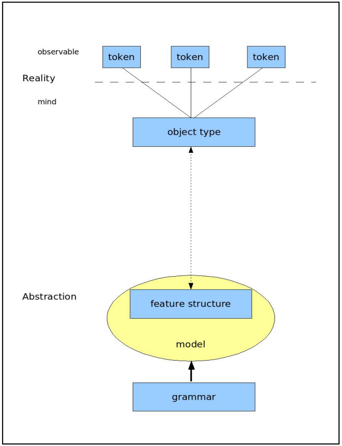
HPSG as in P & S 1994

- structures represent objects that constitute the language of the grammar
- task of the grammar: specifying the **object types** of a natural language
- feature structures are idealizations of equivalence classes of well-formed linguistic entities
- feature structures are not partial, but **complete representations** of linguistic entities
- partiality in grammar only occurs as partial descriptions of complete feature structures

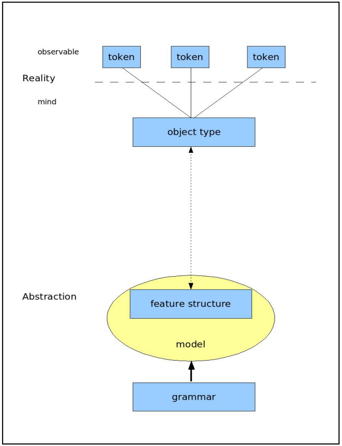
HPSG as in P & S 1994

- structures represent objects that constitute the language of the grammar
- task of the grammar: specifying the **object types** of a natural language
- feature structures are idealizations of equivalence classes of well-formed linguistic entities
- feature structures are not partial, but **complete representations** of linguistic entities
- partiality in grammar only occurs as partial descriptions of complete feature structures
- but: **ontological status** of the structures is subject to dispute

P & S 1994 : Feature structures as object types

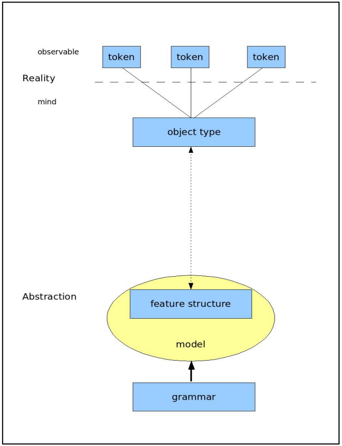


P & S 1994 : Feature structures as object types



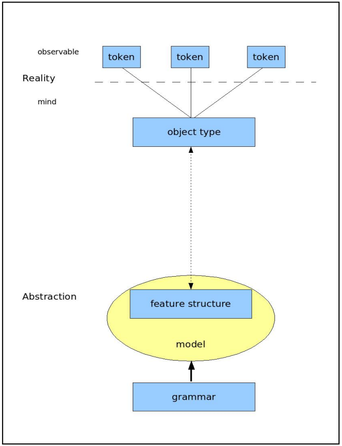
- knowledge of language is knowledge of its object types

P & S 1994 : Feature structures as object types



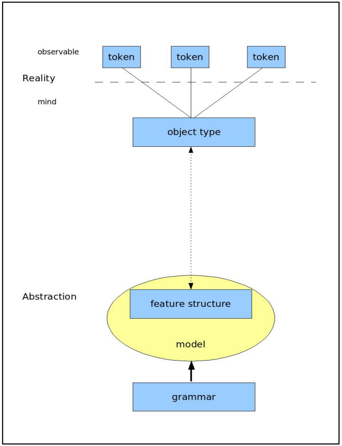
- knowledge of language is knowledge of its object types
- object types are real objects present in the minds of speakers

P & S 1994 : Feature structures as object types



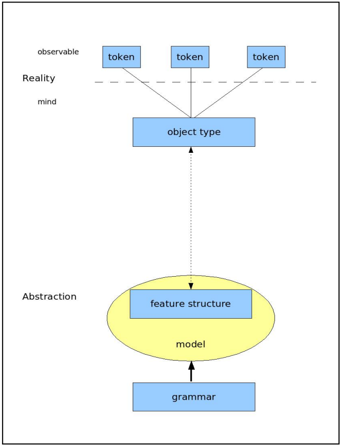
- knowledge of language is knowledge of its object types
- object types are real objects present in the minds of speakers
- a theory of a grammar should include mathematical entities that model object types

P & S 1994 : Feature structures as object types



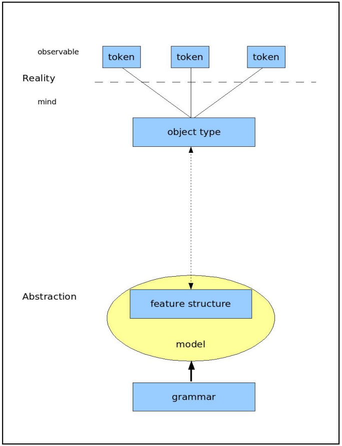
- knowledge of language is knowledge of its object types
- object types are real objects present in the minds of speakers
- a theory of a grammar should include mathematical entities that model object types
- there is a conventional correspondance between token and modeled object type

P & S 1994 : Feature structures as object types



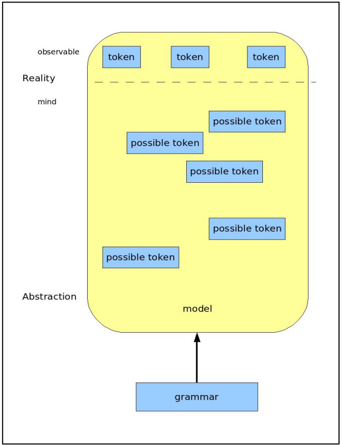
- knowledge of language is knowledge of its object types
- object types are real objects present in the minds of speakers
- a theory of a grammar should include mathematical entities that model object types
- there is a conventional correspondance between token and modeled object type
- linguists have to agree on a correspondence

P & S 1994 : Feature structures as object types

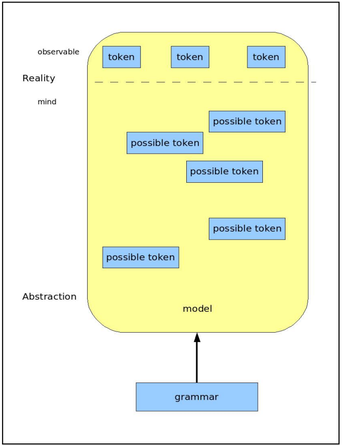


- knowledge of language is knowledge of its object types
- object types are real objects present in the minds of speakers
- a theory of a grammar should include mathematical entities that model object types
- there is a conventional correspondance between token and modeled object type
- linguists have to agree on a correspondence
- if they don't, no falsification is possible

King 1999 : Exhaustive models

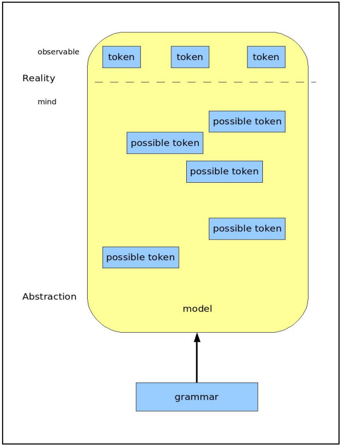


King 1999 : Exhaustive models



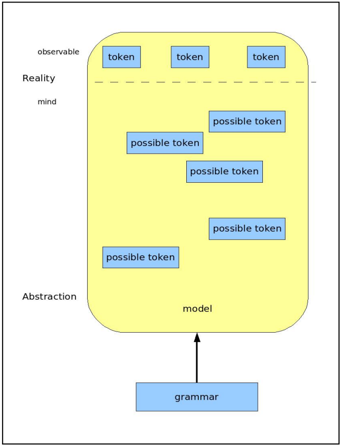
- object types are not a useful concept because there is no evidence for them

King 1999 : Exhaustive models



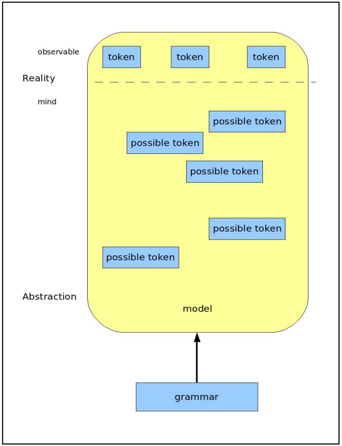
- object types are not a useful concept because there is no evidence for them
- grammar should talk directly about observable data, which are the language tokens

King 1999 : Exhaustive models



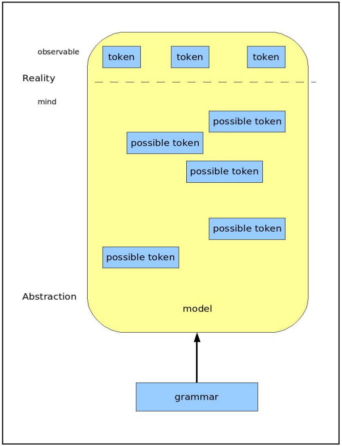
- object types are not a useful concept because there is no evidence for them
- grammar should talk directly about observable data, which are the language tokens
- no intervening mathematical structures between grammar and observable data

King 1999 : Exhaustive models



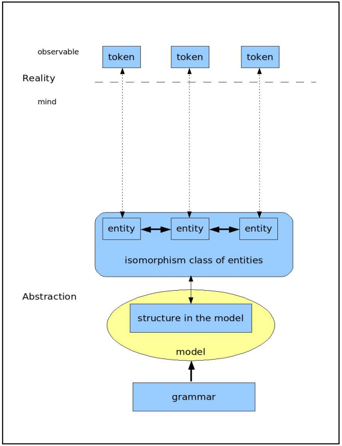
- object types are not a useful concept because there is no evidence for them
- grammar should talk directly about observable data, which are the language tokens
- no intervening mathematical structures between grammar and observable data
- must introduce **possible tokens** that are part of a grammar, but never occur

King 1999 : Exhaustive models

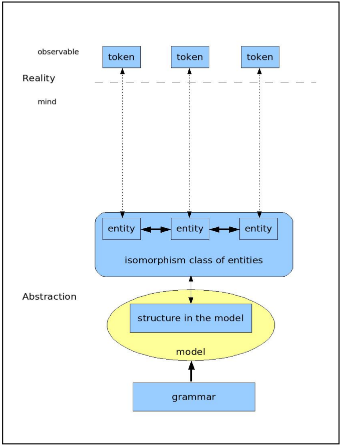


- object types are not a useful concept because there is no evidence for them
- grammar should talk directly about observable data, which are the language tokens
- no intervening mathematical structures between grammar and observable data
- must introduce **possible tokens** that are part of a grammar, but never occur
- bars ways to avoid falsification of a theory by observable data

Pollard 1999 : Strong Generative Capacity

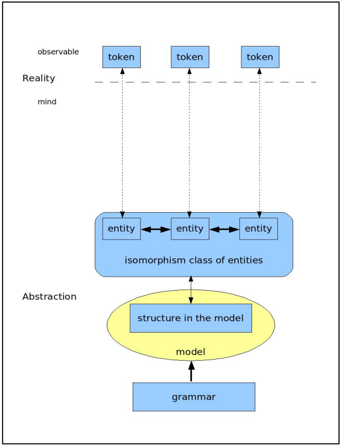


Pollard 1999 : Strong Generative Capacity



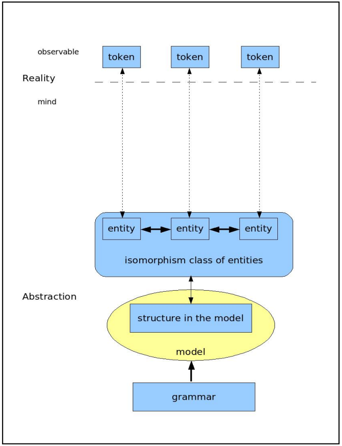
- tightens relation between grammar and data

Pollard 1999 : Strong Generative Capacity



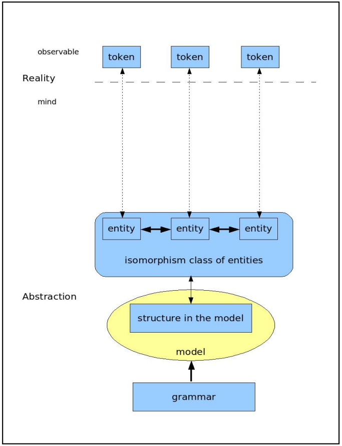
- tightens relation between grammar and data
- conventional correspondance replaced by isomorphism

Pollard 1999 : Strong Generative Capacity



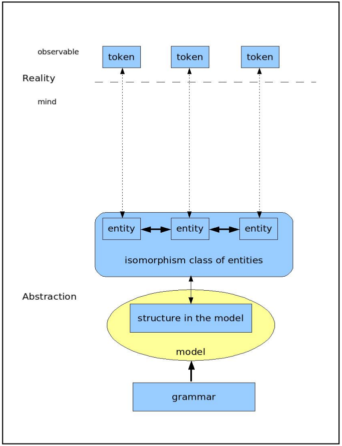
- tightens relation between grammar and data
- conventional correspondance replaced by isomorphism
- entities in the model become isomorphic mathematical idealizatons of concrete tokens

Pollard 1999 : Strong Generative Capacity



- tightens relation between grammar and data
- conventional correspondance replaced by isomorphism
- entities in the model become isomorphic mathematical idealizatons of concrete tokens
- this cannot be done with classical feature structures

Pollard 1999 : Strong Generative Capacity



- tightens relation between grammar and data
- conventional correspondance replaced by isomorphism
- entities in the model become isomorphic mathematical idealizatons of concrete tokens
- this cannot be done with classical feature structures
- insists on an intervening mathematical domain

HPSG 87: Language data as partial information

- typical example of a unification-based or information-based grammar formalism

HPSG 87: Language data as partial information

- typical example of a unification-based or information-based grammar formalism
- unification still the basis of most HPSG implementations

HPSG 87: Language data as partial information

- typical example of a unification-based or information-based grammar formalism
- unification still the basis of most HPSG implementations
- pieces of partial information are assumed to be in a **subsumption hierarchy**

HPSG 87: Language data as partial information

- typical example of a unification-based or information-based grammar formalism
- unification still the basis of most HPSG implementations
- pieces of partial information are assumed to be in a **subsumption hierarchy**
- all possible pieces of information together with the subsumption relation constitute a **Heyting algebra**
- leads to interesting account of language processing

HPSG 87: Language data as partial information

- typical example of a unification-based or information-based grammar formalism
- unification still the basis of most HPSG implementations
- pieces of partial information are assumed to be in a **subsumption hierarchy**
- all possible pieces of information together with the subsumption relation constitute a **Heyting algebra**
- leads to interesting account of language processing
  - hearers accumulate information by unifying pieces of partial information that become available to them from various sources

HPSG 87: Language data as partial information

- typical example of a unification-based or information-based grammar formalism
- unification still the basis of most HPSG implementations
- pieces of partial information are assumed to be in a **subsumption hierarchy**
- all possible pieces of information together with the subsumption relation constitute a **Heyting algebra**
- leads to interesting account of language processing
  - hearers accumulate information by unifying pieces of partial information that become available to them from various sources
  - algebra is not directional, generation and parsing can really be treated as two sides of the same coin

HPSG 87: Language data as partial information

- typical example of a unification-based or information-based grammar formalism
- unification still the basis of most HPSG implementations
- pieces of partial information are assumed to be in a **subsumption hierarchy**
- all possible pieces of information together with the subsumption relation constitute a **Heyting algebra**
- leads to interesting account of language processing
  - hearers accumulate information by unifying pieces of partial information that become available to them from various sources
  - algebra is not directional, generation and parsing can really be treated as two sides of the same coin
- BUT: real understanding of what is going on requires an intuitionistic logic

HPSG 94: Language as a collection of total objects

- a constraint-based or object-based grammar formalism

HPSG 94: Language as a collection of total objects

- a constraint-based or object-based grammar formalism
- has inspired most linguistic work with HPSG so far

HPSG 94: Language as a collection of total objects

- a constraint-based or object-based grammar formalism
- has inspired most linguistic work with HPSG so far
- only totally well-typed and sort resolved entities

HPSG 94: Language as a collection of total objects

- a constraint-based or object-based grammar formalism
- has inspired most linguistic work with HPSG so far
- only totally well-typed and sort resolved entities
- no need for a subsumption relation and a corresponding algebra
- mathematically less complex, but different opinions on

HPSG 94: Language as a collection of total objects

- a constraint-based or object-based grammar formalism
- has inspired most linguistic work with HPSG so far
- only totally well-typed and sort resolved entities
- no need for a subsumption relation and a corresponding algebra
- mathematically less complex, but different opinions on
  - the ontological status of the entities of the interpretations

HPSG 94: Language as a collection of total objects

- a constraint-based or object-based grammar formalism
- has inspired most linguistic work with HPSG so far
- only totally well-typed and sort resolved entities
- no need for a subsumption relation and a corresponding algebra
- mathematically less complex, but different opinions on
  - the ontological status of the entities of the interpretations
  - the view of actual tokens of a language

HPSG 94: Language as a collection of total objects

- a constraint-based or object-based grammar formalism
- has inspired most linguistic work with HPSG so far
- only totally well-typed and sort resolved entities
- no need for a subsumption relation and a corresponding algebra
- mathematically less complex, but different opinions on
  - the ontological status of the entities of the interpretations
  - the view of actual tokens of a language

SRL - Speciate Re-Entrant Logic

- SRL provides a class of formal languages that can be used to describe entities

SRL - Speciate Re-Entrant Logic

- SRL provides a class of formal languages that can be used to describe entities
- each formal language contains a signature providing the non-logical symbols and its interpretations

SRL - Speciate Re-Entrant Logic

- SRL provides a class of formal languages that can be used to describe entities
- each formal language contains a signature providing the non-logical symbols and its interpretations
- fundamental intuition behind SRL:
  each expression of its formal languages is true or false of an entity in an interpretation

SRL - Speciate Re-Entrant Logic

- SRL provides a class of formal languages that can be used to describe entities
- each formal language contains a signature providing the non-logical symbols and its interpretations
- fundamental intuition behind SRL:
  each expression of its formal languages is true or false of an entity in an interpretation
- an expression denotes a set of entities

SRL - Speciate Re-Entrant Logic

- SRL provides a class of formal languages that can be used to describe entities
- each formal language contains a signature providing the non-logical symbols and its interpretations
- fundamental intuition behind SRL:
  each expression of its formal languages is true or false of an entity in an interpretation
- an expression denotes a set of entities

SRL signature

Σ is an **SRL signature** iff

- Σ is a triple $< S, A, F >$ where
- S is a set, the set of **species**,
- A is a set, the set of **attributes**, and
- $F : S \times A \longmapsto P(S)$ is the **appropriateness function**

Interpretation of an SRL signature

I is a Σ interpretation iff

- I is a triple $< U, S, A >$ where
- U is a set, the set of **entities** in the universe,
- $S : U \longmapsto S$ is the **species assignment function**,
- A is the **attribute interpretation function**.

How To Flatten Sort Hierarchies

- SRL signatures do not explicitly contain sort hierarchies

How To Flatten Sort Hierarchies

- SRL signatures do not explicitly contain sort hierarchies
- this is no loss in expressiveness because

How To Flatten Sort Hierarchies

- SRL signatures do not explicitly contain sort hierarchies
- this is no loss in expressiveness because
  - sort hierarchies of HPSG 94 are finite partial orders

How To Flatten Sort Hierarchies

- SRL signatures do not explicitly contain sort hierarchies
- this is no loss in expressiveness because
  - sort hierarchies of HPSG 94 are finite partial orders
  - P & S : if σ_1 is a subsort of σ_2 then $\sigma_1 \subseteq \sigma_2$

How To Flatten Sort Hierarchies

- SRL signatures do not explicitly contain sort hierarchies
- this is no loss in expressiveness because
  - sort hierarchies of HPSG 94 are finite partial orders
  - P & S : if σ_1 is a subsort of σ_2 then $\sigma_1 \subseteq \sigma_2$
  - each entity belongs to exactly one maximally specific sort

How To Flatten Sort Hierarchies

- SRL signatures do not explicitly contain sort hierarchies
- this is no loss in expressiveness because
  - sort hierarchies of HPSG 94 are finite partial orders
  - P & S : if σ_1 is a subsort of σ_2 then $\sigma_1 \subseteq \sigma_2$
  - each entity belongs to exactly one maximally specific sort
  - non-maximal sorts are unions of maximal sorts

How To Flatten Sort Hierarchies

- SRL signatures do not explicitly contain sort hierarchies
- this is no loss in expressiveness because
  - sort hierarchies of HPSG 94 are finite partial orders
  - P & S : if σ_1 is a subsort of σ_2 then $\sigma_1 \subseteq \sigma_2$
  - each entity belongs to exactly one maximally specific sort
  - non-maximal sorts are unions of maximal sorts
  - attribute inheritance enforced by attribute interpretation function

How To Flatten Sort Hierarchies

- SRL signatures do not explicitly contain sort hierarchies
- this is no loss in expressiveness because
  - sort hierarchies of HPSG 94 are finite partial orders
  - P & S : if σ_1 is a subsort of σ_2 then $\sigma_1 \subseteq \sigma_2$
  - each entity belongs to exactly one maximally specific sort
  - non-maximal sorts are unions of maximal sorts
  - attribute inheritance enforced by attribute interpretation function
  - we can give functions to map any sort hierarchy to an SRL signature without loss of information

How To Flatten Sort Hierarchies

- SRL signatures do not explicitly contain sort hierarchies
- this is no loss in expressiveness because
  - sort hierarchies of HPSG 94 are finite partial orders
  - P & S : if σ_1 is a subsort of σ_2 then $\sigma_1 \subseteq \sigma_2$
  - each entity belongs to exactly one maximally specific sort
  - non-maximal sorts are unions of maximal sorts
  - attribute inheritance enforced by attribute interpretation function
  - we can give functions to map any sort hierarchy to an SRL signature without loss of information
- formal languages of SRL can express all aspects of sort hierarchies: to state something about non-maximal sorts, we build a disjunction of all their subspecies

SRL terms and descriptions

In any signature,

- : is a term

SRL terms and descriptions

In any signature,

- : is a term
- a term and an attribute produce another term (a **path**)

SRL terms and descriptions

In any signature,

- : is a term
- a term and an attribute produce another term (a **path**)
- a term $\sim$ a species is a description (a **sort assignment**)

SRL terms and descriptions

In any signature,

- : is a term
- a term and an attribute produce another term (a **path**)
- a term $\sim$ a species is a description (a **sort assignment**)
- a term $\approx$ a term is a description (a **path equation**)

SRL terms and descriptions

In any signature,

- : is a term
- a term and an attribute produce another term (a **path**)
- a term $\sim$ a species is a description (a **sort assignment**)
- a term $\approx$ a term is a description (a **path equation**)
- for a description δ, $\neg\delta$ is again a description

SRL terms and descriptions

In any signature,

- : is a term
- a term and an attribute produce another term (a **path**)
- a term $\sim$ a species is a description (a **sort assignment**)
- a term $\approx$ a term is a description (a **path equation**)
- for a description δ, $\neg\delta$ is again a description
- two descriptions linked with $\wedge$, $\vee$ or $\rightarrow$ give again a description

SRL terms and descriptions

In any signature,

- : is a term
- a term and an attribute produce another term (a **path**)
- a term $\sim$ a species is a description (a **sort assignment**)
- a term $\approx$ a term is a description (a **path equation**)
- for a description δ, $\neg\delta$ is again a description
- two descriptions linked with $\wedge$, $\vee$ or $\rightarrow$ give again a description

SRL term interpretation

- SRL terms denote links between entities in an interpretation

SRL term interpretation

- SRL terms denote links between entities in an interpretation
- SRL descriptions can be seen as denoting sets of entities in an interpretation

SRL term interpretation

- SRL terms denote links between entities in an interpretation
- SRL descriptions can be seen as denoting sets of entities in an interpretation

SRL grammar

Γ is an **SRL grammar** iff

- Γ is a pair $< \Sigma, \theta >$,
- Σ is an SRL signature, and
- θ is a subset of the set of descriptions over Σ

SRL theory denotation function

For each Σ interpretation $\mathsf{I} = < U, S, A >$,

- Θ_I is the total function mapping sets of descriptions to entities, such that for each set of descriptions θ,
- $\Theta_I(\theta) = \{u \in U|$ u is in the interpretation of each description in $\theta\}$

Models in SRL

For each Σ interpretaiton $I =< U, S, A >$,

- I is a Γ **model** iff $\Theta_I(\theta) = U$.

King 1999: Towards Truth in HPSG

- King investigates the question of when an SRL grammar is true of a language

King 1999: Towards Truth in HPSG

- King investigates the question of when an SRL grammar is true of a language
- formulates three necessary conditions for this to hold

King 1999: Towards Truth in HPSG

- King investigates the question of when an SRL grammar is true of a language
- formulates three necessary conditions for this to hold
- these conditions are met if a natural language belongs to the class of exhaustive models of a grammar

King 1999: Towards Truth in HPSG

- King investigates the question of when an SRL grammar is true of a language
- formulates three necessary conditions for this to hold
- these conditions are met if a natural language belongs to the class of exhaustive models of a grammar
- meaning of an SRL grammar can be determined by delineating the class of its exhaustive models

King 1999: Towards Truth in HPSG

- King investigates the question of when an SRL grammar is true of a language
- formulates three necessary conditions for this to hold
- these conditions are met if a natural language belongs to the class of exhaustive models of a grammar
- meaning of an SRL grammar can be determined by delineating the class of its exhaustive models
- directly characterizing language without intervention of a mathematical structure

King 1999: Towards Truth in HPSG

- King investigates the question of when an SRL grammar is true of a language
- formulates three necessary conditions for this to hold
- these conditions are met if a natural language belongs to the class of exhaustive models of a grammar
- meaning of an SRL grammar can be determined by delineating the class of its exhaustive models
- directly characterizing language without intervention of a mathematical structure
- natural languages themselves as intended models of grammars

Why models are not sufficient

- why is our defintion of an SRL grammar insufficient to determine its meaning?

Why models are not sufficient

- why is our defintion of an SRL grammar insufficient to determine its meaning?
- main problem: grammars have multiple models that differ in linguistically relevant ways

Why models are not sufficient

- why is our defintion of an SRL grammar insufficient to determine its meaning?
- main problem: grammars have multiple models that differ in linguistically relevant ways
    - every grammar trivially has a model with the empty universe

Why models are not sufficient

- why is our defintion of an SRL grammar insufficient to determine its meaning?
- main problem: grammars have multiple models that differ in linguistically relevant ways
  - every grammar trivially has a model with the empty universe
  - a valid model of an English grammar could contain only the phrase "Kim likes bagels"

Why models are not sufficient

- why is our defintion of an SRL grammar insufficient to determine its meaning?
- main problem: grammars have multiple models that differ in linguistically relevant ways
  - every grammar trivially has a model with the empty universe
  - a valid model of an English grammar could contain only the phrase “Kim likes bagels”
- any model missing a sentence licensed by the grammar is too small to be the intended model

Why models are not sufficient

- why is our defintion of an SRL grammar insufficient to determine its meaning?
- main problem: grammars have multiple models that differ in linguistically relevant ways
  - every grammar trivially has a model with the empty universe
  - a valid model of an English grammar could contain only the phrase “Kim likes bagels”
- any model missing a sentence licensed by the grammar is too small to be the intended model
- what is missing: a **model theory** that tells us which are the intended models

Why models are not sufficient

- why is our defintion of an SRL grammar insufficient to determine its meaning?
- main problem: grammars have multiple models that differ in linguistically relevant ways
  - every grammar trivially has a model with the empty universe
  - a valid model of an English grammar could contain only the phrase "Kim likes bagels"
- any model missing a sentence licensed by the grammar is too small to be the intended model
- what is missing: a **model theory** that tells us which are the intended models

Overlicensing and Underlicensing

- what we want to know in generative terms is all the structures a grammar **generates**

Overlicensing and Underlicensing

- what we want to know in generative terms is all the structures a grammar **generates**
- in our constraint-based framework, we must instead ask what the grammar **licenses**

Overlicensing and Underlicensing

- what we want to know in generative terms is all the structures a grammar **generates**
- in our constraint-based framework, we must instead ask what the grammar **licenses**
- generative terminology: overgenerating $\Longleftrightarrow$ undergenerating

Overlicensing and Underlicensing

- what we want to know in generative terms is all the structures a grammar **generates**
- in our constraint-based framework, we must instead ask what the grammar **licenses**
- generative terminology: overgenerating $\Longleftrightarrow$ undergenerating
- this can also be called: overlicensing $\Longleftrightarrow$ underlicensing

Overlicensing and Underlicensing

- what we want to know in generative terms is all the structures a grammar **generates**
- in our constraint-based framework, we must instead ask what the grammar **licenses**
- generative terminology: overgenerating $\Longleftrightarrow$ undergenerating
- this can also be called: overlicensing $\Longleftrightarrow$ underlicensing
- so does a given model tell us whether the grammar overlicenses or underlicenses?

Overlicensing and Underlicensing

- what we want to know in generative terms is all the structures a grammar **generates**
- in our constraint-based framework, we must instead ask what the grammar **licenses**
- generative terminology: overgenerating $\Longleftrightarrow$ undergenerating
- this can also be called: overlicensing $\Longleftrightarrow$ underlicensing
- so does a given model tell us whether the grammar overlicenses or underlicenses?

Intuition behind exhaustive models

- we can never be sure

Intuition behind exhaustive models

- we can never be sure
  - if a model contains all the intended structures, there might be a bigger model of the grammar that contains structures not intended: the grammar might overlicense

Intuition behind exhaustive models

- we can never be sure
  - if a model contains all the intended structures, there might be a bigger model of the grammar that contains structures not intended: the grammar might overlicense
  - if a model does not contain all the intended structures, there might be a bigger model that contains all these structures: we cannot tell whether the grammar underlicenses

Intuition behind exhaustive models

- we can never be sure
  - if a model contains all the intended structures, there might be a bigger model of the grammar that contains structures not intended: the grammar might overlicense
  - if a model does not contain all the intended structures, there might be a bigger model that contains all these structures: we cannot tell whether the grammar underlicenses
- the model theory should give us models that tell us whether a grammar overlicenses or underlicenses

Intuition behind exhaustive models

- we can never be sure
  - if a model contains all the intended structures, there might be a bigger model of the grammar that contains structures not intended: the grammar might overlicense
  - if a model does not contain all the intended structures, there might be a bigger model that contains all these structures: we cannot tell whether the grammar underlicenses
- the model theory should give us models that tell us whether a grammar overlicenses or underlicenses
- those models are to be the **exhaustive models** of a grammar

Step I: Components of Entities

An entity u_1 is a component of another entity u_2 iff

- there is a term which in the given interpretation describes u_2 and
- there is a description path leading from this term to u_1

Step II: Interpretation under Entities

An interpretation is the interpretation under an entity u iff

- its universe comprises only all the components of u
- its species assignment function assigns species only to the components of u
- its attribute interpretation function only describes attributes of the components of u

Step II: Interpretation under Entities

An interpretation is the interpretation under an entity u iff

- its universe comprises only all the components of u
- its species assignment function assigns species only to the components of u
- its attribute interpretation function only describes attributes of the components of u

- this can be seen as the subalgebra generated by u in its interpretation

Step III: Subconfigurations of Entities

$< u, I_u >$ is a configuration of entities under an entity u iff

- I_u is the interpretation under u in I

Step III: Subconfigurations of Entities

$< u, I_u >$ is a configuration of entities under an entity u iff

- I_u is the interpretation under u in I

Step IV: SRL Congruence between Configurations

Two configurations are **SRL congruent** iff

- there is a bijection between the components of both configurations that
  - assigns to each component a component of equal species
  - lets each component have the same attributes as its counterpart
  - maps the values of those attributes to their counterparts

Step V: Conditions for a Grammar

An SRL grammar is true of a natural language only if

1. the natural language can be seen as an interpretation of the grammar's signature
2. this interpretation is a model of the grammar
3. any entity of another interpretation for which no entity in the model has a isomorphic configuration does not fulfill one of the descriptions in the grammar

Step V: Conditions for a Grammar

An SRL grammar is true of a natural language only if

1. the natural language can be seen as an interpretation of the grammar's signature
2. this interpretation is a model of the grammar
3. any entity of another interpretation for which no entity in the model has a isomorphic configuration does not fulfill one of the descriptions in the grammar

- condition 1 ties intended interpretations to the signature

Step V: Conditions for a Grammar

An SRL grammar is true of a natural language only if

1. the natural language can be seen as an interpretation of the grammar's signature
2. this interpretation is a model of the grammar
3. any entity of another interpretation for which no entity in the model has a isomorphic configuration does not fulfill one of the descriptions in the grammar

- condition 1 ties intended interpretations to the signature
- condition 2 ties intended interpretations to the theory

Step V: Conditions for a Grammar

An SRL grammar is true of a natural language only if

1. the natural language can be seen as an interpretation of the grammar's signature
2. this interpretation is a model of the grammar
3. any entity of another interpretation for which no entity in the model has a isomorphic configuration does not fulfill one of the descriptions in the grammar

- condition 1 ties intended interpretations to the signature
- condition 2 ties intended interpretations to the theory
- condition 3 says the model contains all **possible tokens** of the language and is thus an **exhaustive model**

Step VI: Simulation of Interpretations

- why do we have a class of exhaustive models?

Step VI: Simulation of Interpretations

- why do we have a class of exhaustive models?
  - linguistic entities have unknown mathematical properties

Step VI: Simulation of Interpretations

- why do we have a class of exhaustive models?
  - linguistic entities have unknown mathematical properties
  - but to say something about them, we need mathematical structures

Step VI: Simulation of Interpretations

- why do we have a class of exhaustive models?
    - linguistic entities have unknown mathematical properties
    - but to say something about them, we need mathematical structures
    - since we cannot claim to know much about the needed structures, we resort to a class of models defined independently of the linguistic nature of their entities

Step VI: Simulation of Interpretations

- why do we have a class of exhaustive models?
  - linguistic entities have unknown mathematical properties
  - but to say something about them, we need mathematical structures
  - since we cannot claim to know much about the needed structures, we resort to a class of models defined independently of the linguistic nature of their entities
  - any model in that class may then be used in an investigation of the descriptive properties of the language

Step VI: Simulation of Interpretations

- why do we have a class of exhaustive models?
  - linguistic entities have unknown mathematical properties
  - but to say something about them, we need mathematical structures
  - since we cannot claim to know much about the needed structures, we resort to a class of models defined independently of the linguistic nature of their entities
  - any model in that class may then be used in an investigation of the descriptive properties of the language
- to define this class, we need a notion of **simulation**:

Step VI: Simulation of Interpretations

- why do we have a class of exhaustive models?
  - linguistic entities have unknown mathematical properties
  - but to say something about them, we need mathematical structures
  - since we cannot claim to know much about the needed structures, we resort to a class of models defined independently of the linguistic nature of their entities
  - any model in that class may then be used in an investigation of the descriptive properties of the language
- to define this class, we need a notion of **simulation**:

An interpretation simulates another interpretation iff

- for each entity in one interpretation, the configuration under this entity has a SRL congruent counterpart in the other interpretation

Step VII: Exhaustive Models

An interpretation is an **exhaustive model** iff

- it is a model of the grammar and
- it simulates every other model of the grammar

- for every configuration under an entity in any other model of the grammar, we find an SRL congruent counterpart in I

Step VIII: Existence of Exhaustive Models

Theorem

For each SRL signature Σ, for each Σ theory θ, there exists a Σ interpretation I such that I is an exhaustive $< \Sigma, \theta >$ model.

- this theorem allows us to explain the meaning of an arbitrary SRL grammar in terms of its exhaustive models

The original ideas of Pollard & Sag 1994

- an HPSG 94 grammar is about the object types of a language, not about possible tokens

The original ideas of Pollard & Sag 1994

- an HPSG 94 grammar is about the object types of a language, not about possible tokens
- some abstraction from tokens to object types is implied

The original ideas of Pollard & Sag 1994

- an HPSG 94 grammar is about the object types of a language, not about possible tokens
- some abstraction from tokens to object types is implied
- object types modeled by **totally well-typed** and **sort resolved** feature structures

The original ideas of Pollard & Sag 1994

- an HPSG 94 grammar is about the object types of a language, not about possible tokens
- some abstraction from tokens to object types is implied
- object types modeled by **totally well-typed** and **sort resolved** feature structures
- an admission function assigns each grammar a set of abstract feature structures

The original ideas of Pollard & Sag 1994

- an HPSG 94 grammar is about the object types of a language, not about possible tokens
- some abstraction from tokens to object types is implied
- object types modeled by **totally well-typed** and **sort resolved** feature structures
- an admission function assigns each grammar a set of abstract feature structures

How to link this to SRL

- losing the distinction between indiscernible possible tokens and grouping them together to classes represented by object types makes the intuitive difference between P & S and King

How to link this to SRL

- losing the distinction between indiscernible possible tokens and grouping them together to classes represented by object types makes the intuitive difference between P & S and King
- abstract feature structures correspond to the object types of natural language

How to link this to SRL

- losing the distinction between indiscernible possible tokens and grouping them together to classes represented by object types makes the intuitive difference between P & S and King
- abstract feature structures correspond to the object types of natural language
- system of possible tokens then corresponds to a collection of concrete feature structures

How to link this to SRL

- losing the distinction between indiscernible possible tokens and grouping them together to classes represented by object types makes the intuitive difference between P & S and King
- abstract feature structures correspond to the object types of natural language
- system of possible tokens then corresponds to a collection of concrete feature structures
- this means: object types can be seen as equivalence classes of tokens

Relation of 94 concrete feature structures to SRL entities

- 94 concrete feature structures are defined in ways similar to finite state automata

Relation of 94 concrete feature structures to SRL entities

- 94 concrete feature structures are defined in ways similar to finite state automata
  - components are seen as nodes/states, with the root as initial state

Relation of 94 concrete feature structures to SRL entities

- 94 concrete feature structures are defined in ways similar to finite state automata
  - components are seen as nodes/states, with the root as initial state
  - moving down into the structure along a path is like making transitions in an automaton

Relation of 94 concrete feature structures to SRL entities

- 94 concrete feature structures are defined in ways similar to finite state automata
  - components are seen as nodes/states, with the root as initial state
  - moving down into the structure along a path is like making transitions in an automaton
  - the presence of attributes and the values assigned to them can be encoded like a transition function

Relation of 94 concrete feature structures to SRL entities

- 94 concrete feature structures are defined in ways similar to finite state automata
  - components are seen as nodes/states, with the root as initial state
  - moving down into the structure along a path is like making transitions in an automaton
  - the presence of attributes and the values assigned to them can be encoded like a transition function
- for each entity in any interpretation of an SRL signature, there is a concrete feature structure with that entity as its root node

Relation of 94 concrete feature structures to SRL entities

- 94 concrete feature structures are defined in ways similar to finite state automata
  - components are seen as nodes/states, with the root as initial state
  - moving down into the structure along a path is like making transitions in an automaton
  - the presence of attributes and the values assigned to them can be encoded like a transition function
- for each entity in any interpretation of an SRL signature, there is a concrete feature structure with that entity as its root node
- a 94 feature structure determined by I with root node u is the same as the configuration under u in I

Relation of 94 concrete feature structures to SRL entities

- 94 concrete feature structures are defined in ways similar to finite state automata
  - components are seen as nodes/states, with the root as initial state
  - moving down into the structure along a path is like making transitions in an automaton
  - the presence of attributes and the values assigned to them can be encoded like a transition function
- for each entity in any interpretation of an SRL signature, there is a concrete feature structure with that entity as its root node
- a 94 feature structure determined by I with root node u is the same as the configuration under u in I
- SRL congruence can be seen as CFS equivalence with different node names

From 94 CFS to object types

- isomorphic concrete feature structures have different nodes and cannot be collapsed

From 94 CFS to object types

- isomorphic concrete feature structures have different nodes and cannot be collapsed
- an abstraction function maps isomorphic 94 CFS to the same 94 abstract feature structure

From 94 CFS to object types

- isomorphic concrete feature structures have different nodes and cannot be collapsed
- an abstraction function maps isomorphic 94 CFS to the same 94 abstract feature structure
- King's exhaustive models only differ in the number of isomorphic 94 CFS in a given shape

From 94 CFS to object types

- isomorphic concrete feature structures have different nodes and cannot be collapsed
- an abstraction function maps isomorphic 94 CFS to the same 94 abstract feature structure
- King's exhaustive models only differ in the number of isomorphic 94 CFS in a given shape
- the set of abstract feature structures admitted by a grammar is basically equivalent to one of its exhaustive models

From 94 CFS to object types

- isomorphic concrete feature structures have different nodes and cannot be collapsed
- an abstraction function maps isomorphic 94 CFS to the same 94 abstract feature structure
- King's exhaustive models only differ in the number of isomorphic 94 CFS in a given shape
- the set of abstract feature structures admitted by a grammar is basically equivalent to one of its exhaustive models
- each object type is an equivalence class of indiscernible possible tokens

Pollard 1999: SGC in HPSG

- strictly representational approach

Pollard 1999: SGC in HPSG

- strictly representational approach
- no claims about status of object types, redefines them as isomorphism classes of structures that include idealized tokens

Pollard 1999: SGC in HPSG

- strictly representational approach
- no claims about status of object types, redefines them as isomorphism classes of structures that include idealized tokens
- modeling structures are no longer classical feature structures

Pollard 1999: SGC in HPSG

- strictly representational approach
- no claims about status of object types, redefines them as isomorphism classes of structures that include idealized tokens
- modeling structures are no longer classical feature structures
- Pollard's goal: a precise explanation in which sense an HPSG grammar is a generative grammar

Pollard 1999: SGC in HPSG

- strictly representational approach
- no claims about status of object types, redefines them as isomorphism classes of structures that include idealized tokens
- modeling structures are no longer classical feature structures
- Pollard's goal: a precise explanation in which sense an HPSG grammar is a generative grammar
- that means: formal definition of the **Strong Generative Capacity** of a grammar.

Intuitions about the SGC

- no two members are structurally isomorphic

Intuitions about the SGC

- no two members are structurally isomorphic
- if the grammar is correct, exactly those tokens structurally isomorphic to entities in the SGC will be judged grammatical

Intuitions about the SGC

- no two members are structurally isomorphic
- if the grammar is correct, exactly those tokens structurally isomorphic to entities in the SGC will be judged grammatical
- relation to types: there is only one representative of each class of isomorphically configured linguistic entities

Intuitions about the SGC

- no two members are structurally isomorphic
- if the grammar is correct, exactly those tokens structurally isomorphic to entities in the SGC will be judged grammatical
- relation to types: there is only one representative of each class of isomorphically configured linguistic entities
- relation to tokens: each token structurally isomorphic to an entity of the SGC

Intuitions about the SGC

- no two members are structurally isomorphic
- if the grammar is correct, exactly those tokens structurally isomorphic to entities in the SGC will be judged grammatical
- relation to types: there is only one representative of each class of isomorphically configured linguistic entities
- relation to tokens: each token structurally isomorphic to an entity of the SGC

Step I: Pollard Feature Structures

A **Pollard feature structure** determined by u in I is

- the interpretation containing all entities that are "accessible" from u

Step II: Pollard Abstract Feature Structures

An **Pollard abstract feature structure** is

- a set of isomorphic Pollard feature structure fed into a node abstraction that constructs equivalence classes of entities

Step III: Strong Generative Capacity

For each SRL signature Σ, the SGC is

- the total function from grammars to classes of Pollard abstract feature structures over Σ such that
- for each theory consisting of SRL descriptions over Σ,
- the abstract feature structures in the respective SGC comprise only those that are abstractions of entities in some interpretation of the grammar and that are discernable from each other because not isomorphically structured

Parallels to the other theories

- SGC can be defined starting from abstract feature structures (as we did), from exhaustive models or from a notion of generation that again relies on the abstract feature structures modeled by the grammar
- the SGC must also be an exhaustive model of the grammar
- the abstraction step makes it similar to a collection of object types

Conclusion: Linking the approaches

- there are different views on the meaning of HPSG grammars that differ in philosophically significant ways

Conclusion: Linking the approaches

- there are different views on the meaning of HPSG grammars that differ in philosophically significant ways
- different traditions in philosophy lead to different model theories

Conclusion: Linking the approaches

- there are different views on the meaning of HPSG grammars that differ in philosophically significant ways
- different traditions in philosophy lead to different model theories
- however, it is possible to bring those views together by mathematical means

Conclusion: Linking the approaches

- there are different views on the meaning of HPSG grammars that differ in philosophically significant ways
- different traditions in philosophy lead to different model theories
- however, it is possible to bring those views together by mathematical means
- the views turn out to be interchangeable for our purposes since they do not interfere with what we are actually doing with the grammars

Conclusion: Linking the approaches

- there are different views on the meaning of HPSG grammars that differ in philosophically significant ways
- different traditions in philosophy lead to different model theories
- however, it is possible to bring those views together by mathematical means
- the views turn out to be interchangeable for our purposes since they do not interfere with what we are actually doing with the grammars
- however, it is still useful to know something about the background

Conclusion: Linking the approaches

- there are different views on the meaning of HPSG grammars that differ in philosophically significant ways
- different traditions in philosophy lead to different model theories
- however, it is possible to bring those views together by mathematical means
- the views turn out to be interchangeable for our purposes since they do not interfere with what we are actually doing with the grammars
- however, it is still useful to know something about the background
- understanding at least one of the theories helps answer the most urgent questions about the meaning of the feature structures we are dealing with each day

References

- King 1999: Towards Truth in HPSG
- Pollard & Sag 1994: Head Driven Phrase Structure Grammar
- Pollard 1999: Strong Generative Capacity in HPSG
- Richter 2001: A Mathematical Formalism for HPSG
- Richter 2004: A Mathematical Formalism for Linguistic Theories with an Application in HPSG
- Richter 2006: A Web-based Course in Grammar Formalisms and Parsing

Thank you.