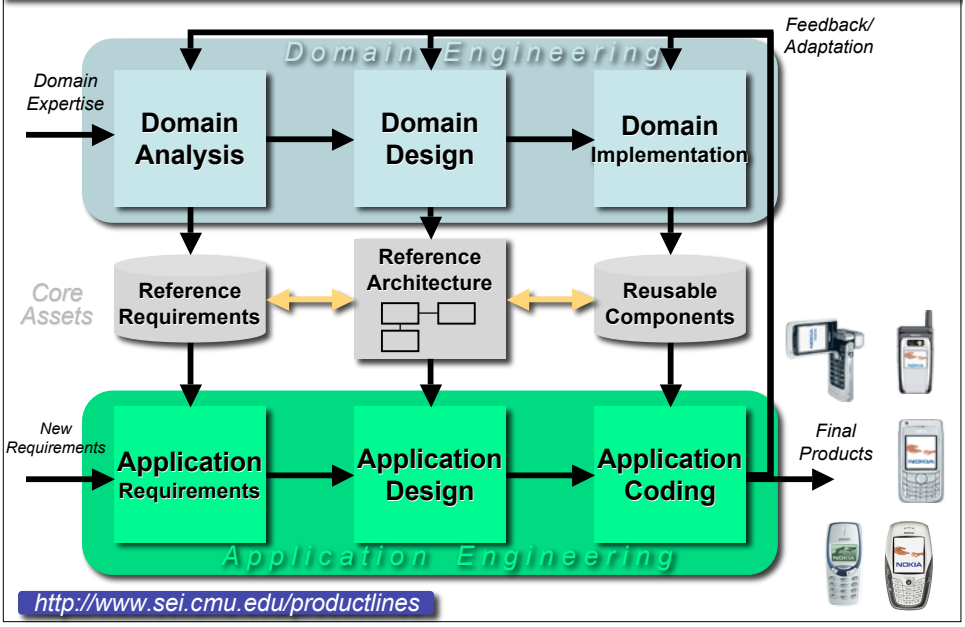
Overview

- Background
- Research question
- Research method
- Applying the method
- Main findings
- Summary of contributions
- Limitations and threats to validity
- Work in progress

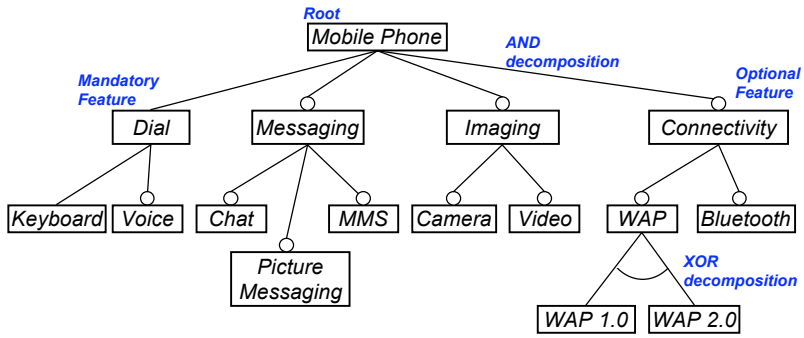
Overview

- **Background**
- Research question
- Research method
- Applying the method
- Main findings
- Summary of contributions
- Limitations and threats to validity
- Work in progress

Software Product Lines (SPL)



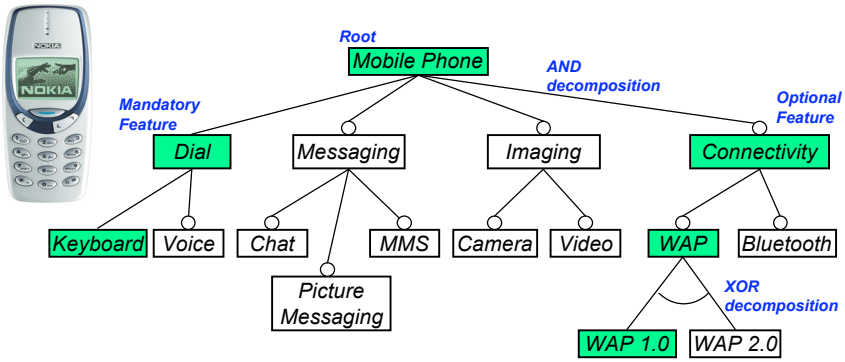
Modeling coarse-grained variability with Feature Diagrams (FD)



Additional constraint(s):
Picture Messaging **`requires`** *Camera*

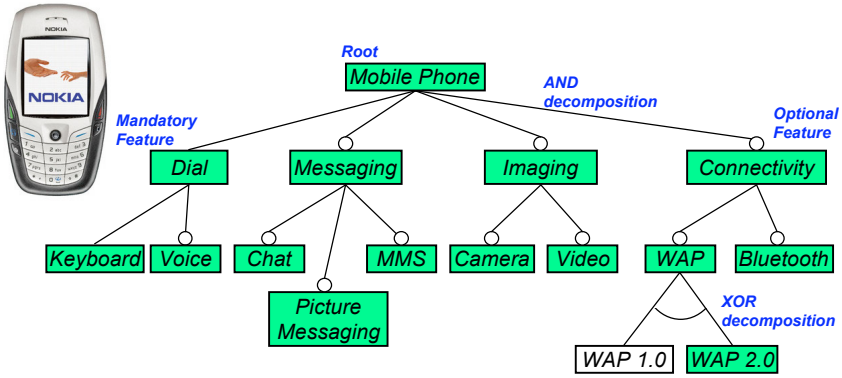
FODA: Feature-Oriented Domain Analysis [Kang *et al.*, 1990]

Modeling coarse-grained variability with Feature Diagrams (FD)



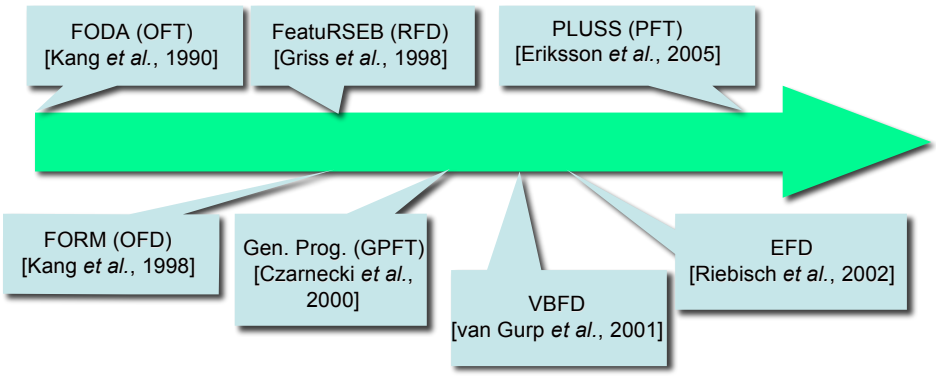
Additional constraint(s):
Picture Messaging **`requires`** *Camera*

Modeling coarse-grained variability with Feature Diagrams (FD)

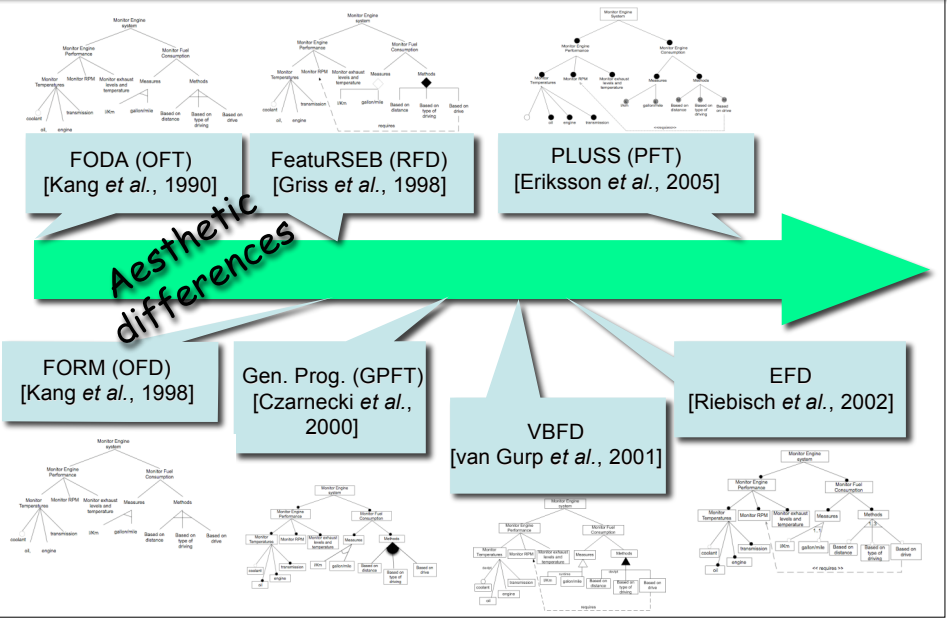


Additional constraint(s):
Picture Messaging **`requires`** *Camera*

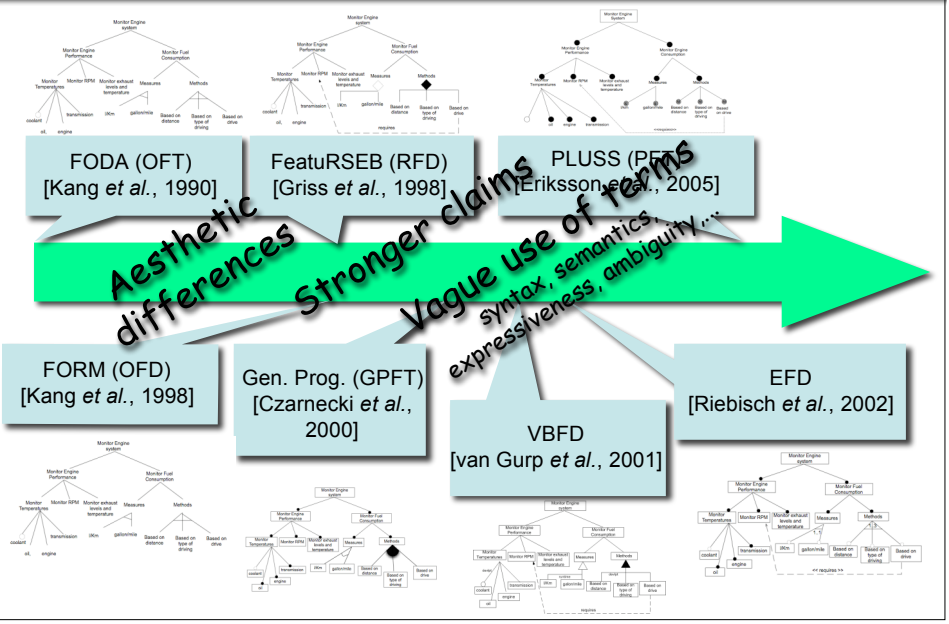
Existing FD languages



Existing FD languages



Existing FD languages



Overview

- Background
- **Research question**
- Research method
- Applying the method
- Main findings
- Summary of contributions
- Limitations and threats to validity
- Work in progress

Research question

- General: which notations should one use to model SPL variability at a coarse-grained level?
- Specific: how do existing FD notations evaluate and compare wrt **formal notions/criteria**?
  - syntax & semantics
  - ambiguity
  - expressiveness
  - embeddability
  - succinctness
  - complexity of decision procedures

Why are those criteria important?

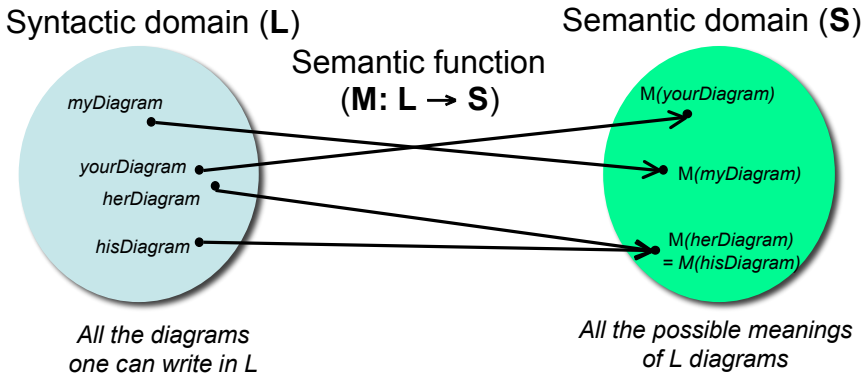
- For *research*
  - to collect **objective** knowledge on FD languages
  - to improve them **usefully**
  - to elaborate **safe** and **efficient reasoning** mechanisms
- For *practice*
  - to accelerate the advent of a **standard well-defined FD language**
  - to deliver **powerful tools**
  - to **decrease complexity** of SPL variability management

Overview

- Background
- Research question
- **Research method**
- Applying the method
- Main findings
- Summary of contributions
- Limitations and threats to validity
- Work in progress

Formal semantics

[Harel & Rumpe, *Meaningful modeling*, IEEE Computer, 2004]



- A language is (formally) **ambiguity-free** when
  - L, S and M all receive a **mathematical definition**
  - M is a **function**!
  - M is **complete**, i.e. defined for all d ∈ L

Overview

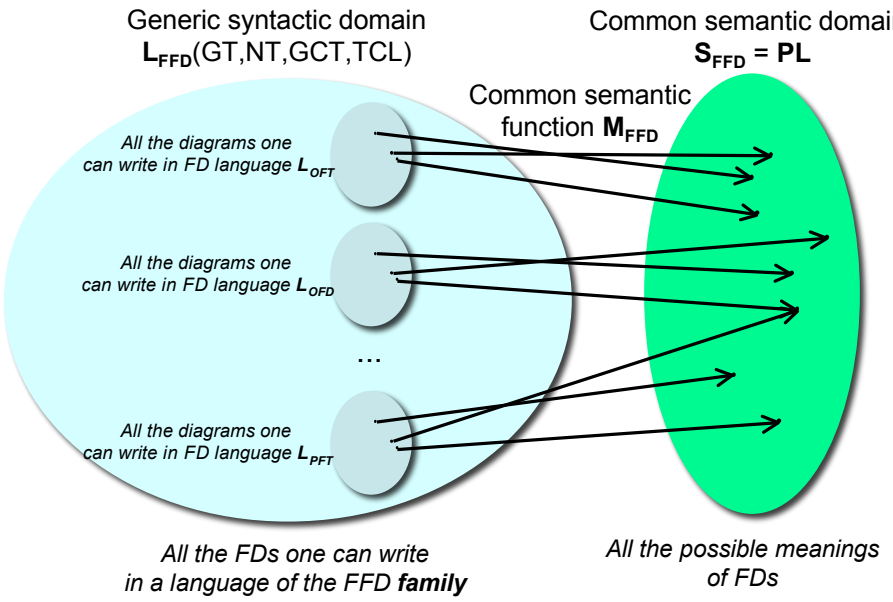
- Background
- Research question
- Research method
- **Applying the method**
- Main findings
- Summary of contributions
- Limitations and threats to validity
- Work in progress

Applying the method

- Formal definition of FODA [Bontemps *et al.*, ICFI, 2004]
- General formal definition of FODA-inspired languages [Schobbens *et al.*, RE, 2006]
  - generic abstract syntax: $\mathbf{L_{FFD}}$
    - ignoring *aesthetic* differences between the languages (concrete syntaxes)
    - sensitive to the differences (parameters) that have an impact on the semantics
  - formal semantics : $\mathbf{S_{FFD}}$ and $\mathbf{M_{FFD}}$
    - defined only once for all FODA-inspired languages
- Study formal properties

  [Schobbens *et al.*, J. Computer Networks, 2007]

Applying the method

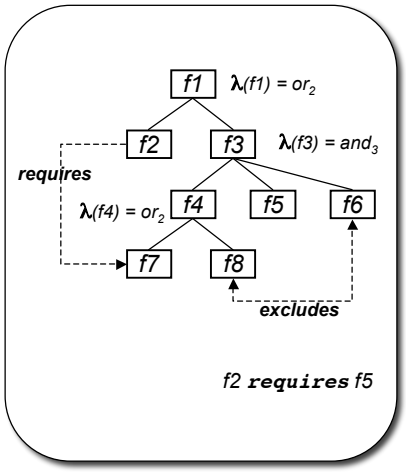


The L_{FFD} generic abstract syntax

Every FD **d** ∈ $\mathbf{L_{FFD}}$**(GT,NT,GCT,TCL)** is a tuple **(N,r, λ,DE,CE,Φ)** such that

- **N** *the set of node*
- **r** ∈ **N** *the root*
- **DE** ⊆ **N** × **N** *the decomposition edges (obeying GT)*
- **λ : N→NT** *the function labelling nodes with boolean operators*
- **CE** ⊆ **N** × **GCT** × **N** *the constraint edges*
- **Φ** ⊆ **TCL** *the textual constraints*

- + 4 well-formedness constraints e.g. *only r has no parent*

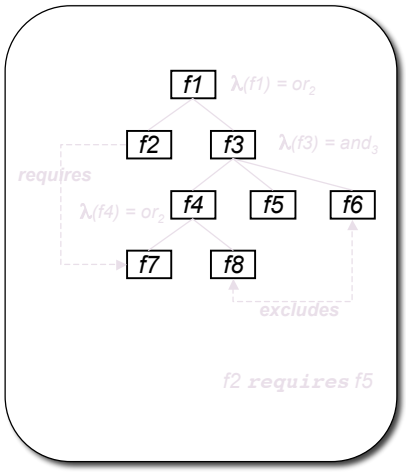


The L_{FFD} generic abstract syntax

Every FD **d** ∈ $\mathbf{L_{FFD}}$**(GT,NT,GCT,TCL)** is a tuple **(N,r, λ,DE,CE,Φ)** such that

- **N** *the set of nodes*
- **r** ∈ **N** *the root*
- **DE** ⊆ **N** × **N** *the decomposition edges (obeying GT)*
- **λ : N→NT** *the function labelling nodes with boolean operators*
- **CE** ⊆ **N** × **GCT** × **N** *the constraint edges*
- **Φ** ⊆ **TCL** *the textual constraints*

- + 4 well-formedness constraints e.g. *only r has no parent*

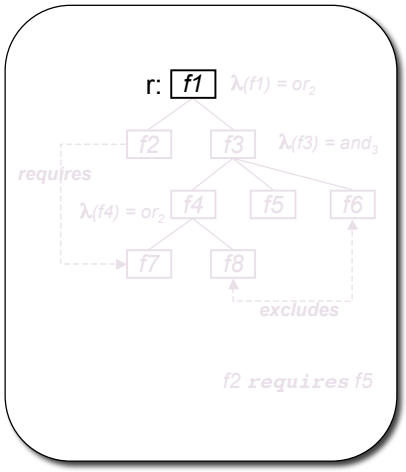


The L_{FFD} generic abstract syntax

Every FD **d** ∈ $\mathbf{L_{FFD}}$**(GT,NT,GCT,TCL)** is a tuple **(N,r, λ,DE,CE,Φ)** such that

- **N** *the set of nodes*
- **r ∈ N** *the root*
- **DE ⊆ N × N** *the decomposition edges (obeying GT)*
- **λ : N→NT** *the function labelling nodes with boolean operators*
- **CE ⊆ N × GCT × N** *the constraint edges*
- **Φ ⊆ TCL** *the textual constraints*

- + 4 well-formedness constraints e.g. *only r has no parent*

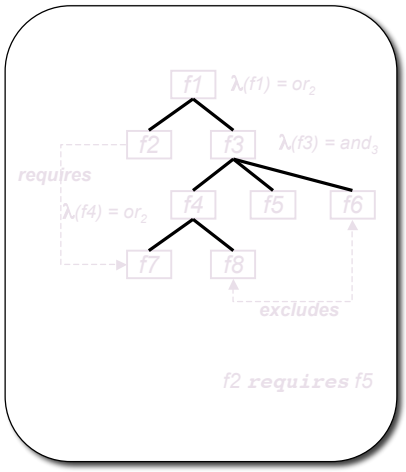


The L_{FFD} generic abstract syntax

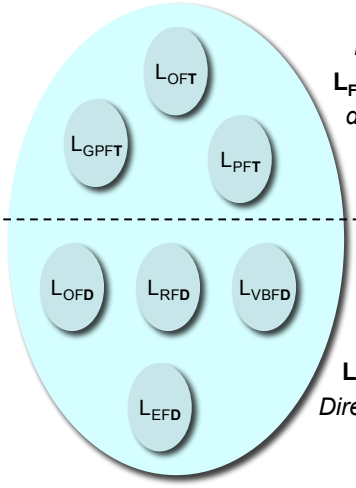
Every FD **d** ∈ $\mathbf{L_{FFD}}$**(GT,NT,GCT,TCL)** is a tuple **(N,r, λ,DE,CE,Φ)** such that

- **N** *the set of nodes*
- **r ∈ N** *the root*
- **DE ⊆ N × N** *the decomposition edges (obeying **GT**)*
- **λ : N→NT** *the function labelling nodes with boolean operators*
- **CE ⊆ N × GCT × N** *the constraint edges*
- **Φ ⊆ TCL** *the textual constraints*

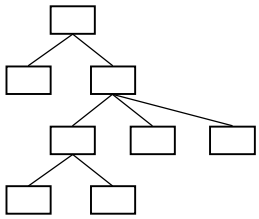
- + 4 well-formedness constraints e.g. *only r has no parent*



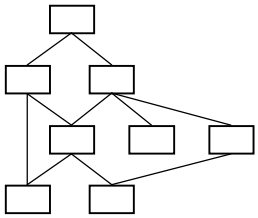
Languages differ in graph type



For languages in $\mathbf{L_{FFD}}$**(TREE**, _ , _ , _)
diagrams can only be tree-shaped



For languages in $\mathbf{L_{FFD}}$**(DAG**, _ , _ , _)
Directed Acyclic Graphs are allowed too

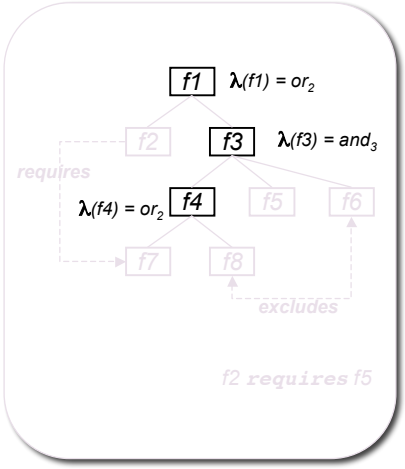


The L_{FFD} generic abstract syntax

Every FD **d** ∈ $\mathbf{L_{FFD}}$**(GT,NT,GCT,TCL)** is a tuple **(N,r, λ,DE,CE,Φ)** such that

- **N** *the set of nodes*
- **r** ∈ **N** *the root*
- **DE** ⊆ **N** × **N** *the decomposition edges (obeying GT)*
- **λ : N→NT** *the function labelling nodes with boolean operators*
- **CE** ⊆ **N** × **GCT** × **N** *the constraint edges*
- **Φ** ⊆ **TCL** *the textual constraints*

- + 4 well-formedness constraints
  e.g. *only r has no parent*

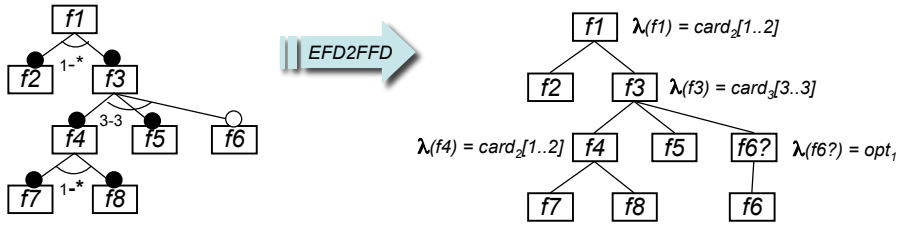


Conventionally, for a *leaf* node n, λ(n) =and_0

Encoding optional nodes and cardinalities

C_{EFD}
(concrete syntax of EFD)

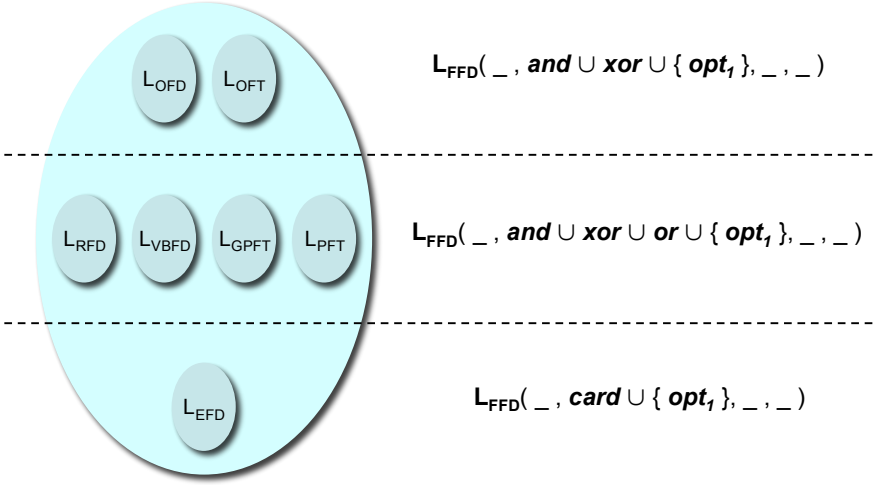
L_{EFD}
$= L_{FFD}$(DAG, {card∪{opt_1}}, {requires,mutex}, CR)



Where
- $card_z[x..y]$ is the operator that returns TRUE if at least x and at most y of its z arguments are TRUE
- opt_z is the operator that always returns TRUE

Cardinalities in FDs were introduced by [Riebisch *et al.*,2002]

Languages differ in node types



L_{FFD}(_ , ***and*** ∪ ***xor*** ∪ { opt_1 }, _ , _)

L_{FFD}(_ , ***and*** ∪ ***xor*** ∪ ***or*** ∪ { opt_1 }, _ , _)

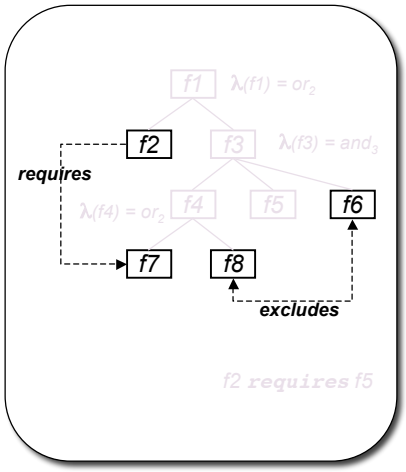
L_{FFD}(_ , ***card*** ∪ { opt_1 }, _ , _)

The L_{FFD} generic abstract syntax

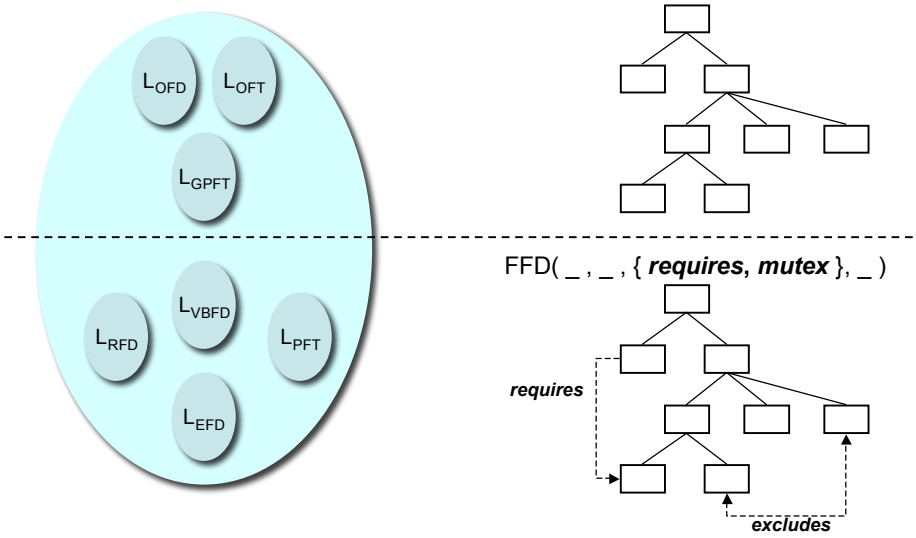
Every FD **d** ∈ $\mathbf{L_{FFD}}$**(GT,NT,GCT,TCL)** is a tuple **(N,r, λ,DE,CE,Φ)** such that

- **N** *the set of nodes*
- **r** ∈ **N** *the root*
- **DE** ⊆ **N** × **N** *the decomposition edges (obeying GT)*
- **λ : N→NT** *the function labelling nodes with boolean operators*
- **CE** ⊆ **N** × **GCT** × **N** *the constraint edges*
- **Φ** ⊆ **TCL** *the textual constraints*

- + 4 well-formedness constraints e.g. *only r has no parent*



Languages differ in graphical constraint types

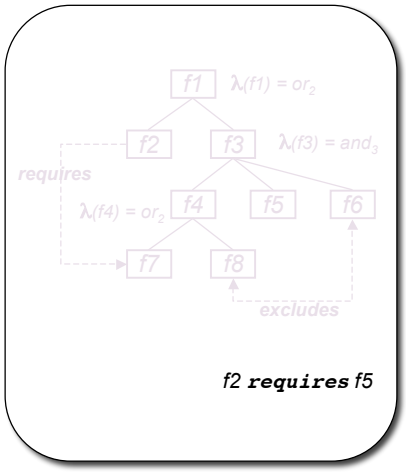


The L_{FFD} generic abstract syntax

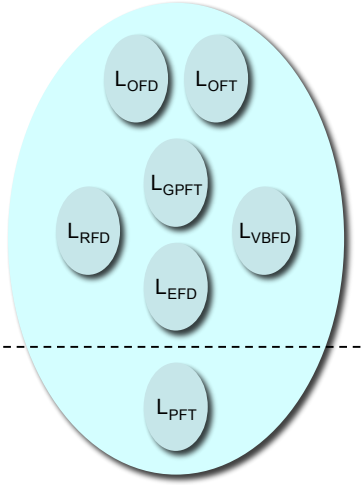
Every FD **d** ∈ $\mathbf{L_{FFD}}$**(GT,NT,GCT,TCL)** is a tuple **(N,r, λ,DE,CE,Φ)** such that

- **N** *the set of nodes*
- **r** ∈ **N** *the root*
- **DE** ⊆ **N** × **N** *the decomposition edges (obeying GT)*
- **λ : N→NT** *the function labelling nodes with boolean operators*
- **CE** ⊆ **N** × **GCT** × **N** *the constraint edges*
- **Φ** ⊆ **TCL** *the textual constraints*

- + 4 well-formedness constraints e.g. *only r has no parent*



Languages differ in textual constraint (sub)language



FFD(_ , _ , _ , **CR**)

where
CR is the language of *composition rules*

CR ::= p1 (***requires*** | ***mutex***) p2

[Kang *et al.*, 1990]

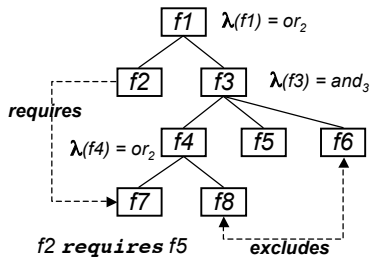
FFD(_ , _ , _ , ∅)

Overview of FD abstract syntaxes

	GT	NT	GCT	TCL
FODA (OFT)	TREE	*and* ∪ *xor* ∪ {opt_1}	∅	CR
FORM (OFD)	DAG	*and* ∪ *xor* ∪ {opt_1}	∅	CR
FeatuRSEB (RFD)	DAG	*and* ∪ *xor* ∪ *or* ∪ {opt_1}	*requires*, *mutex*	CR
van Gurp et al. (VBFD)	DAG	*and* ∪ *xor* ∪ *or* ∪ {opt_1}	*requires*, *mutex*	CR
Riebisch et al. (EFD)	DAG	*card* ∪ {opt_1}	*requires*, *mutex*	CR
Gen. Prog. (GPFT)	TREE	*and* ∪ *xor* ∪ *or* ∪ {opt_1}	∅	CR
PLUSS (PFT)	TREE	*and* ∪ *xor* ∪ *or* ∪ {opt_1}	*requires*, *mutex*	∅

Semantic domain

- A **configuration** is a set of nodes/features
- A **product** is a set of primitive nodes/features
- A **product line** (PL) is a set of products

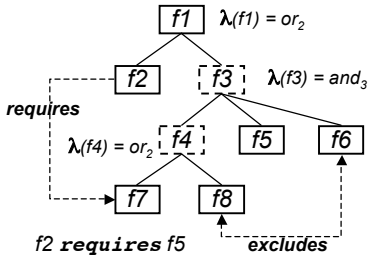


f1 f3 f4 f5 f6 f7 ∈ ℘ N

Semantic domain

- A **configuration** is a set of nodes/features
- A **product** is a set of primitive nodes/features
- A **product line** (PL) is a set of products

E.g. if *f3* and *f4* are « irrelevant » for the stakeholders



removing the non-primitive features

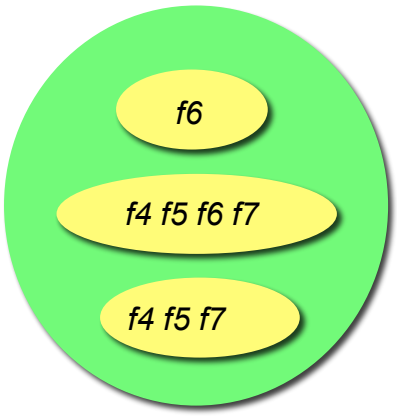
f1 f3 f4 f5 f6 f7 $\in \wp$ N

f1 f5 f6 f7 $\in \wp$ P

where P $\subseteq$ N

Semantic domain

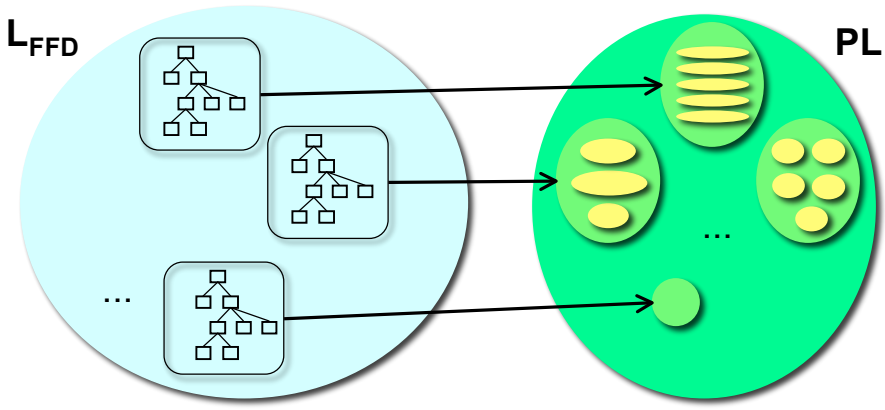
- A **configuration** is a set of nodes/features
- A **product** is a set of primitive nodes/features
- A **product line** (**PL**) is a set of products



$\in S_{FFD}$ = PL = $\wp\ \wp$ P

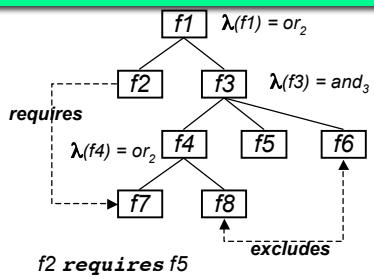
Semantic function's signature

- The **semantic function** (M_{FFD}: $L_{FFD} \rightarrow PL$) associates a PL to every FD

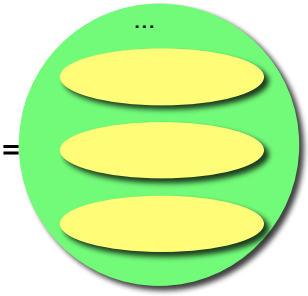


Semantic function's definition

- Definition: for every $d \in L_{FFD}$, $M'_{FFD}(d)$ is the PL such that
  - r (the root) is in every configuration
  - the meaning of the nodes is satisfied
  - Φ (textual constraints) are satisfied
  - CE (graphical constraints) are satisfied
  - if a node $s \neq r$ is in the configuration, one of its parents must be too (*justification* rule)

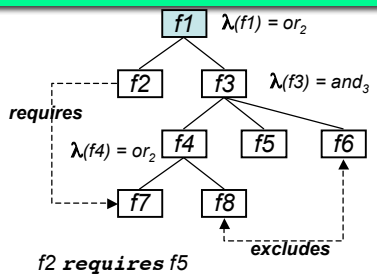


$M'_{FFD}(d) =$

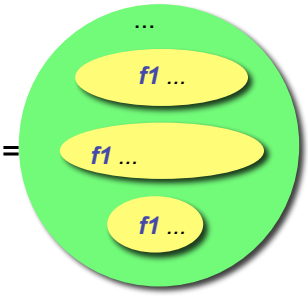


Semantic function's definition

- Definition: for every d ∈ L_{FFD}, $M'_{FFD}(d)$ is the PL such that
  - r (the root) is in every configuration
  - the meaning of the nodes is satisfied
  - Φ (textual constraints) are satisfied
  - CE (graphical constraints) are satisfied
  - if a node s ≠ r is in the configuration, one of its parents must be too (*justification* rule)

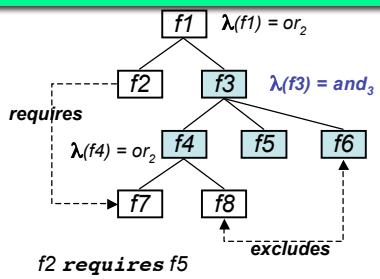


$M'_{FFD}(d)$ =

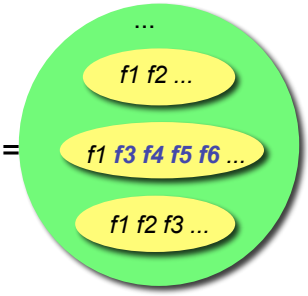


Semantic function's definition

- Definition: for every d ∈ L_{FFD}, $M'_{FFD}(d)$ is the PL such that
  - r (the root) is in every configuration
  - the meaning of the nodes is satisfied
  - Φ (textual constraints) are satisfied
  - CE (graphical constraints) are satisfied
  - if a node s ≠ r is in the configuration, one of its parents must be too (*justification* rule)

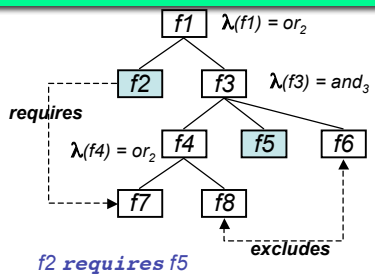


$M'_{FFD}(d)$ =

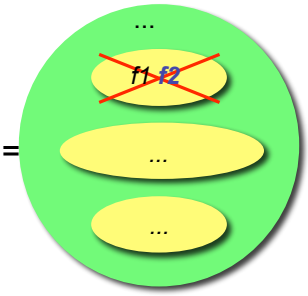


Semantic function's definition

- Definition: for every d ∈ L_{FFD}, $M'_{FFD}(d)$ is the PL such that
  - r (the root) is in every configuration
  - the meaning of the nodes is satisfied
  - Φ (textual constraints) are satisfied
  - CE (graphical constraints) are satisfied
  - if a node s ≠ r is in the configuration, one of its parents must be too (*justification* rule)

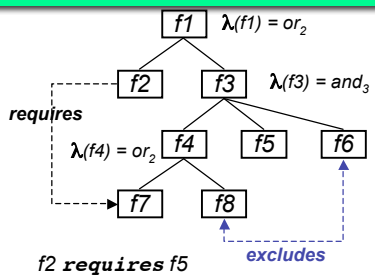


$M'_{FFD}(d)$ =

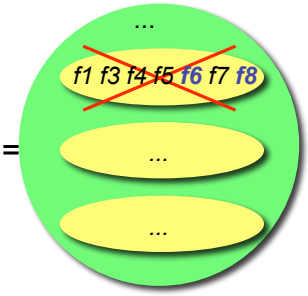


Semantic function's definition

- Definition: for every d ∈ L_{FFD}, $M'_{FFD}(d)$ is the PL such that
  - r (the root) is in every configuration
  - the meaning of the nodes is satisfied
  - Φ (textual constraints) are satisfied
  - CE (graphical constraints) are satisfied
  - if a node s ≠ r is in the configuration, one of its parents must be too (*justification* rule)

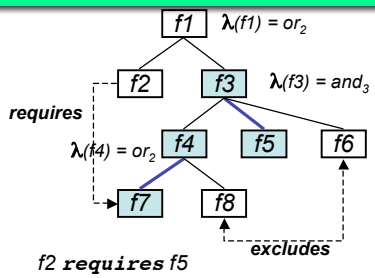


$M'_{FFD}(d)$ =

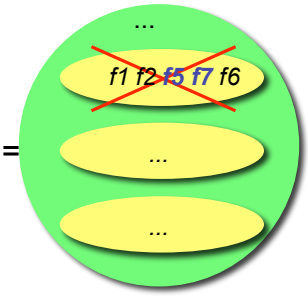


Semantic function's definition

- Definition: for every d ∈ L_{FFD}, $M'_{FFD}(d)$ is the PL such that
  - r (the root) is in every configuration
  - the meaning of the nodes is satisfied
  - Φ (textual constraints) are satisfied
  - CE (graphical constraints) are satisfied
  - if a node s (≠ r) is in the configuration, one of its parents must be too (*justification* rule)

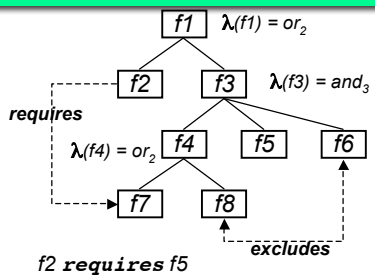


$M'_{FFD}(d)$ =

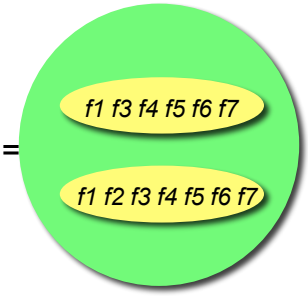


Semantic function's definition

- Definition: for every d ∈ L_{FFD}, $M'_{FFD}(d)$ is the PL such that
  - r (the root) is in every configuration
  - the meaning of the nodes is satisfied
  - Φ (textual constraints) are satisfied
  - CE (graphical constraints) are satisfied
  - if a node s ≠ r is in the configuration, one of its parents must be too (*justification* rule)

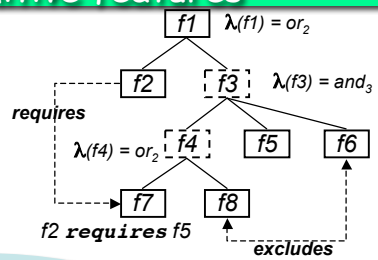


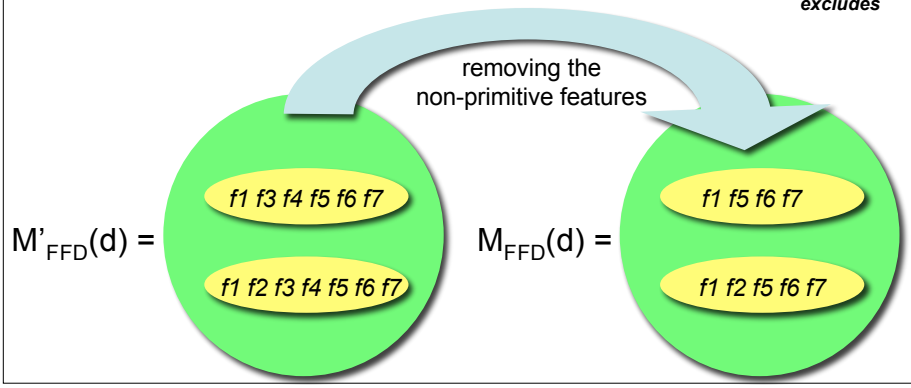
$M'_{FFD}(d)$ =



Semantic function's definition

Dealing with non-primitive features





Overview

- Background
- Research question
- Research method
- Applying the method
- **Main findings**
- Summary of contributions
- Limitations and threats to validity
- Work in progress

Ambiguity

- FODA (OFT) was **not** ambiguous
- Other languages were ambiguous
  - e.g. « *All mandatory features are part of all [products]* »

    EFD — [Riebisch *et al.*,2002]
- We aligned the semantics on FODA
- **Finding**: extensions have not made things clearer
- **Open question**: is our semantics the *intended* one?

Decision Problems and Complexity

- **Satisfiability**: $M_{FFD}(d) \neq \emptyset$
  - NP-complete
  - linear for many tree-shaped graphs
- **Product checking**: $\{f_1,\ldots,f_n\} \in M_{FFD}(d)$
  - NP-complete
  - linear for many tree-shaped graphs, or if P = N
- **Equivalence**: $M_{FFD}(d_1) = M_{FFD}(d_2)$
  - Π_1-complete
- ...
- **Conclusions**:
  - tractability issues but efficient algorithms in many common cases
  - existing algorithms can now be proved for correctness and optimality

[Schobbens *et al.*, J. Computer Networks, 2007]

Other Decision Problems

- **Dead features**: $P \setminus \cup M_{FFD}(d)$
- **Commonalities**: $\cap M_{FFD}(d)$
- **« Merging »** FDs: construct d_3 from d_1 and d_2 s.t.
  - **Intersection**: $M_{FFD}(d_3) = M_{FFD}(d_1) \cap M_{FFD}(d_2)$
  - **Union**: $M_{FFD}(d_3) = M_{FFD}(d_1) \cup M_{FFD}(d_2)$
  - **Reduced product**:
    $M_{FFD}(d_3) = \{p_1 \cup p_2 : p_1 \in M_{FFD}(d_1), p_2 \in M_{FFD}(d_2)\}$
- etc...

[Benavides *et al.*, JISBD, 2006]

Expressiveness

- **Expressiveness**: part of the semantic domain that a language is able to express
- **Expressive completeness**: expressiveness = PL
- **Findings** (ignoring textual and graphical constraints):
  - OF**T**, GPF**T**, PF**T** are ***not*** expressively complete
  - OF**D**, RF**D**, VBF**D,** EF**D** are expressively complete
    - adding *or*-decomposition (e.g. « RFT », « VBFT ») is not enough
- **Conclusion**: extensions of FORM (OFD) [Kang et al., 1998] deal with **readability**

[Schobbens *et al.*, J. Computer Networks, 2007]

Embeddability and Succinctness

- L_1 is **embeddable** in L_2 iff
  - L_2 is as expressive as L_1
  - transformation from L_1 to L_2 preserves structure / is *natural*
- If $L_2 \subset L_1$ and L_1 is embeddable in L_2, L_1 is **harmfully redundant**
- **Conclusion**: a language with only *card* operators is sufficient (wrt expressiveness) and *natural*

$opt_1 \Rightarrow card_1[0..1]$

$xor_s \Rightarrow card_s[1..1]$

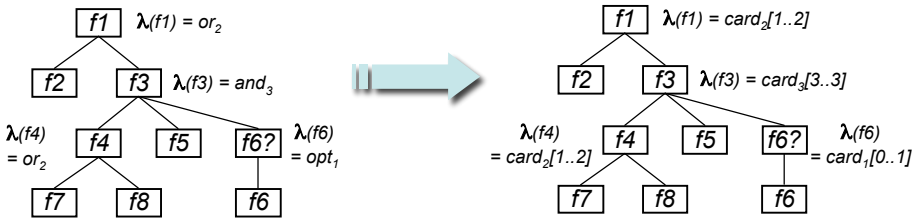
$or_s \Rightarrow card_s[1..s]$

$and_s \Rightarrow card_s[s..s]$

[Schobbens *et al.*, J. Computer Networks, 2007]

VFD

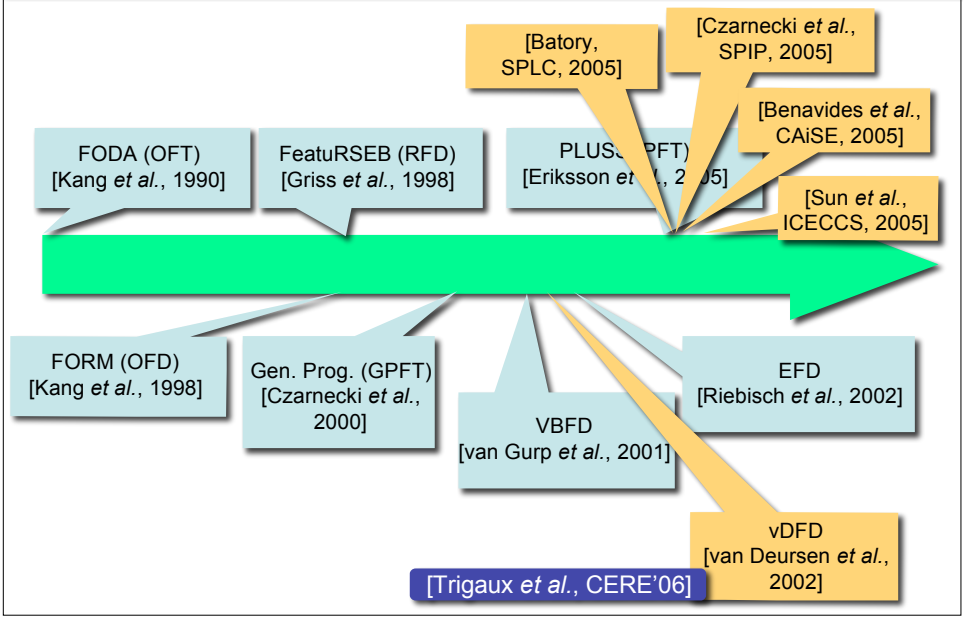
$$L_{VFD} = L_{FFD}(\text{DAG}, \textit{card}, \varnothing, \varnothing)$$



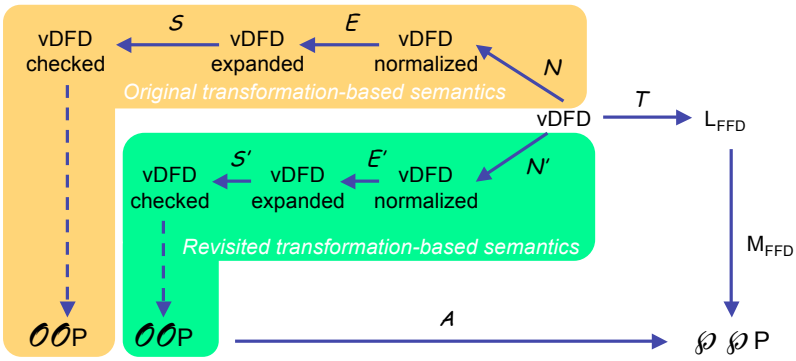
- **Suggestion**: VFD as pivot abstract syntax for tools
  - complete expressiveness
  - **traceability !**

[Schobbens *et al.*, J. Computer Networks, 2007]

Comparative semantics
Dealing with other formal semantics



Comparative semantics
Dealing with other formal semantics



- **Conclusions**:
  - tool-based semantics can be complex and error-prone
    - errors found in transformation rules (defined on concrete syntax)
    - semantic domain is too rich (?)
  - restricted forms of DAGs can also yield expressive completeness

[Trigaux et al., CERE'06]

Overview

- Background
- Reseach question
- Research method
- Applying the method
- Main findings
- **Summary of contributions**
- Threats to validity
- Work in progress

Summary of contributions

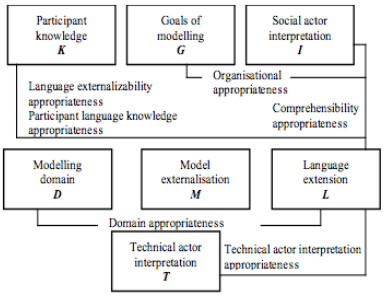
- The **diversity** of FD languages
  - is mostly motivated by **readability** issues
  - **overlooked fundamental notions** (from formal language theory)
- Improved **understanding** and **definition** of FDs
- Opened way for **more objective comparison** of FD (and other?) languages
  - including a **general comparative semantics** method
- Opened way for **efficient** and **safe tool support**

Overview

- Background
- Reseach question
- Research method
- Applying the method
- Main findings
- Summary of contributions
- **Limitations and threats to validity**
- Work in progress

Limitations and threats to validity

- Formal properties are **not the only criteria**



SEQUAL [Krogstie, 2003]

- Our formal semantics ≟ the *best* formal semantics
- More empirical studies needed, e.g. [Djebbi&Salinesi, CERE'06]
- Visual aspects equally important [Moody, REFSQ'06]

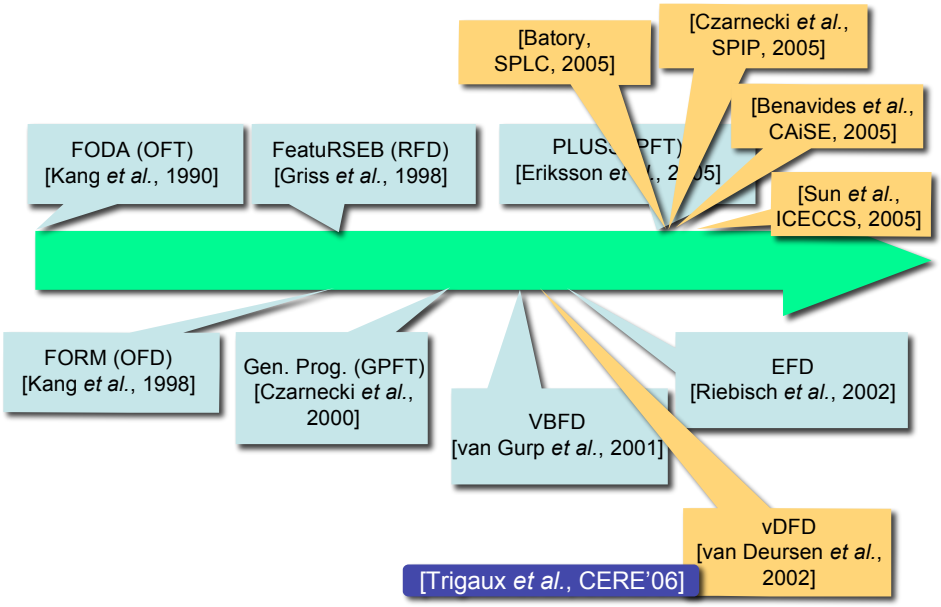
Limitations and threats to validity

- To *formalize* often means to *choose*
  - Possible errors in **interpreting** informal language definitions (both for syntax and semantics)
  - Interpreting FODA (OFT) was straightforward
- Advanced constructs not formalized
  - *binding times*
  - feature *specialisation*, *implementation*, etc.
  - *layers*
  - *feature attributes*
  - *...*

Overview

- Background
- Reseach question
- Research method
- Applying the method
- Main findings
- Summary of contributions
- Threats to validity
- **Work in progress**

Work in progress
Comparative semantics



Work in progress

- Implementing decision procedures
  - VFD as pivot language (translations to/from other languages)
  - mainly SAT-based + specific algorithms for optimizations
- Separation of concerns and co-evolution of FDs
  - separate ***PL*** and ***platform*** variability in distinct but related models

    [Metzger *et al.*, submitted]
- Relating FDs and Jackson's Problem Frames (PF)
  - PFs provide problem-oriented **modularization**
  - PFs focus on **commonality**
  - more precise link between features and
    - **requirements**, i.e. *optative* statements about « real world » (RW)
    - **domain hypotheses**, i.e. *indicative* statements about RW
    - **specifications**, i.e. *optative* black-box description of product behaviour
  - feature interaction detection [Classen *et al.*, VaMoS'07]

Thank you! Any questions?