Like Alice in Wonderland: Unraveling Reasoning and Cognition Using Analogies and Concept Blending

Tarek R. Besold

KRDB, Faculty of Computer Science, Free University of Bozen-Bolzano

16. June 2016

unibz.it

Honour to whom honour is due...

The following is joint work with many people, most notably:

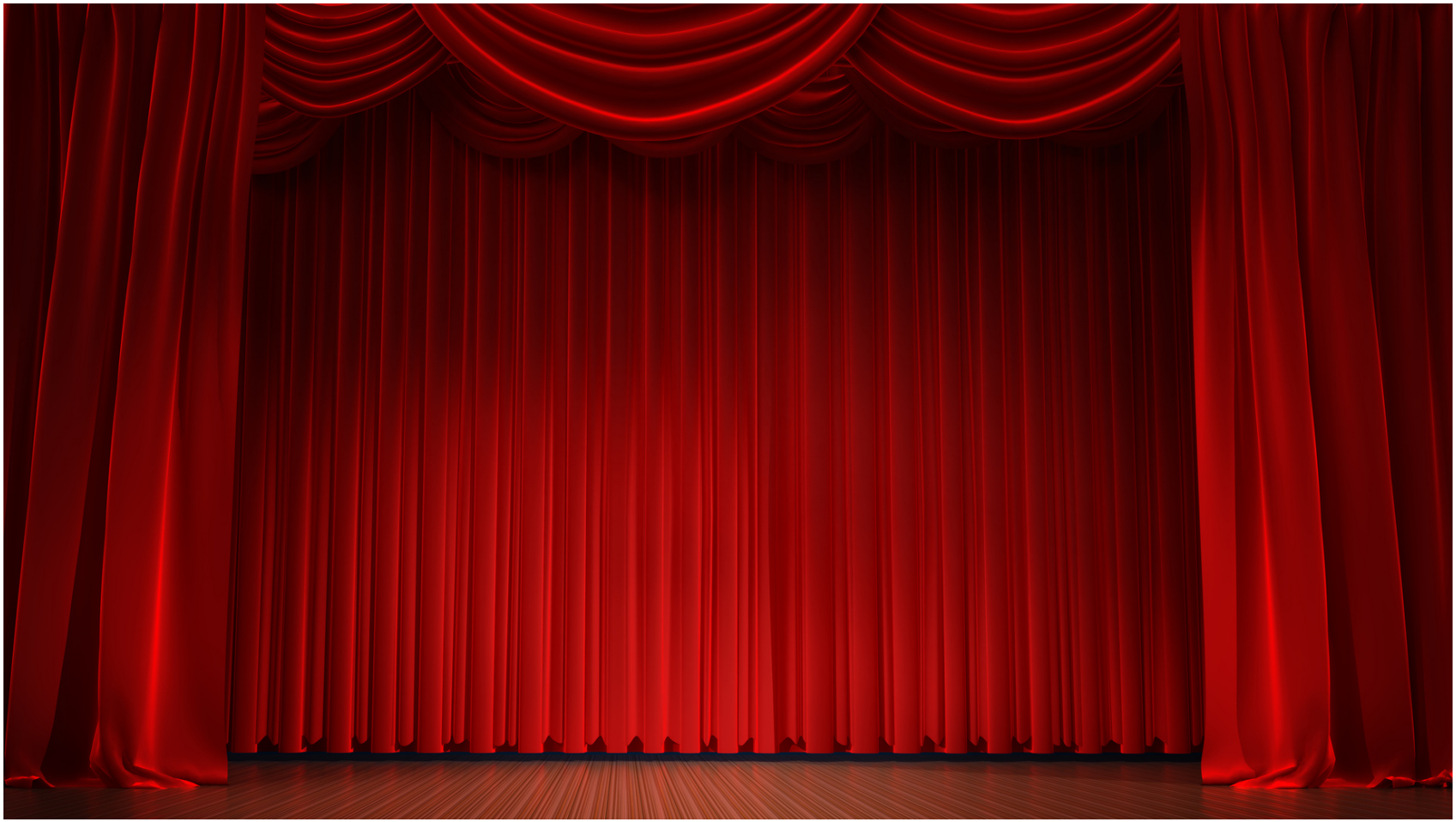
- **Robert Robere**, Department of Computer Science, University of Toronto (Canada).
- **Enric Plaza**, IIIA-CSIC, Barcelona (Spain).
- **Kai-Uwe Kühnberger**, Institute of Cognitive Science, University of Osnabrück (Germany).

And Who is Paying (Some of) the Bills?

The described work on concept blending has been conducted as part of the European FP7 **Concept Invention Theory (COINVENT)** project (FET-Open grant number 611553).

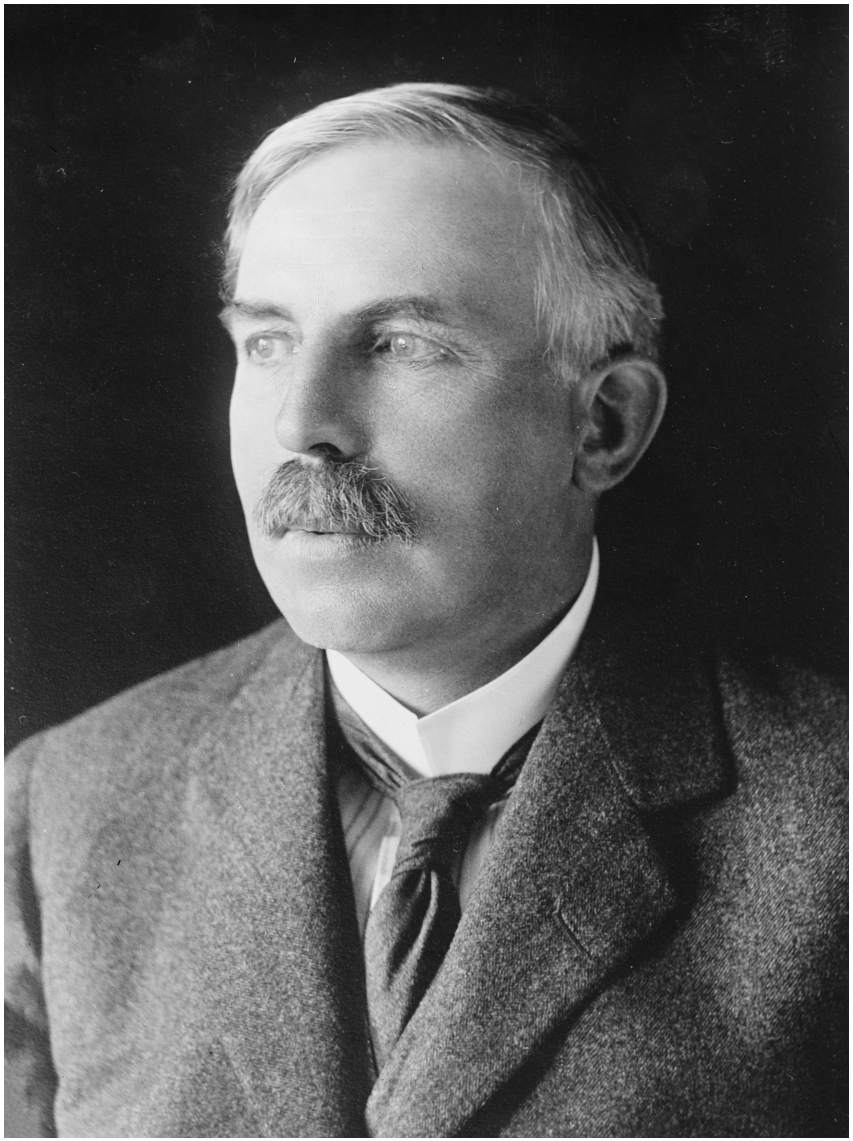
Consortium members are:

- Free University of Bozen-Bolzano (Südtirol-Alto Adige, Italy)
- University of Osnabrück (Germany)
- University of Magdeburg (Germany)
- University of Dundee (Scotland, UK)
- University of Edinburgh (Scotland, UK)
- Goldsmiths, University of London (UK)
- IIIA-CSIC, Barcelona (Catalunya, Spain)
- Aristotle University of Thessaloniki (Greece)

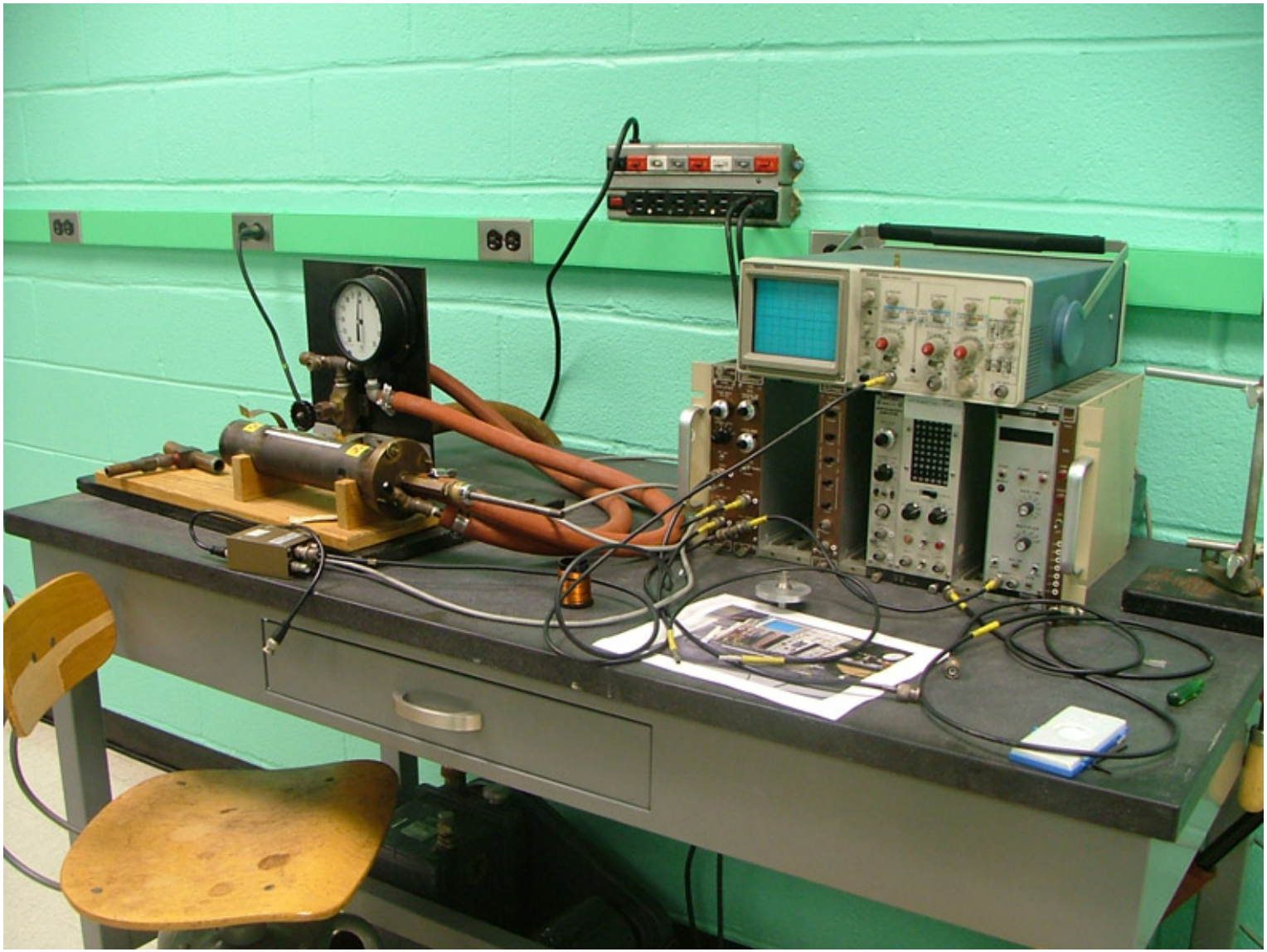


Non-Classical and Cross-Domain Reasoning

Back in the Day (1)

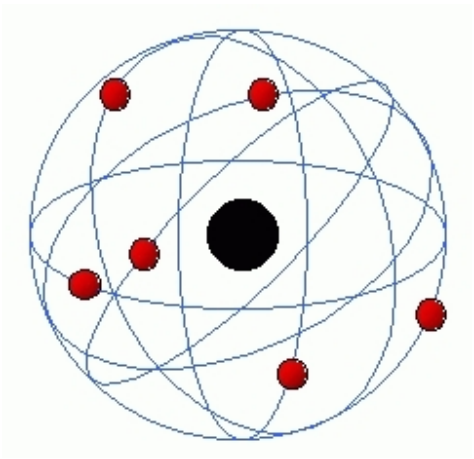


Back in the Day (2)



Back in the Day (3)

Rutherford analogy (underlying the Bohr-Rutherford model of the atom):



- Analogy between solar system and hydrogen atom:
- ...nucleus is more massive than electrons, sun is more massive than planets.
- ...nucleus attracts electrons (Coulomb's law), sun attracts planets (Newton's law of gravity).
- ...attraction plus mass relation causes electrons to revolve around nucleus, similarly planets revolve around sun.

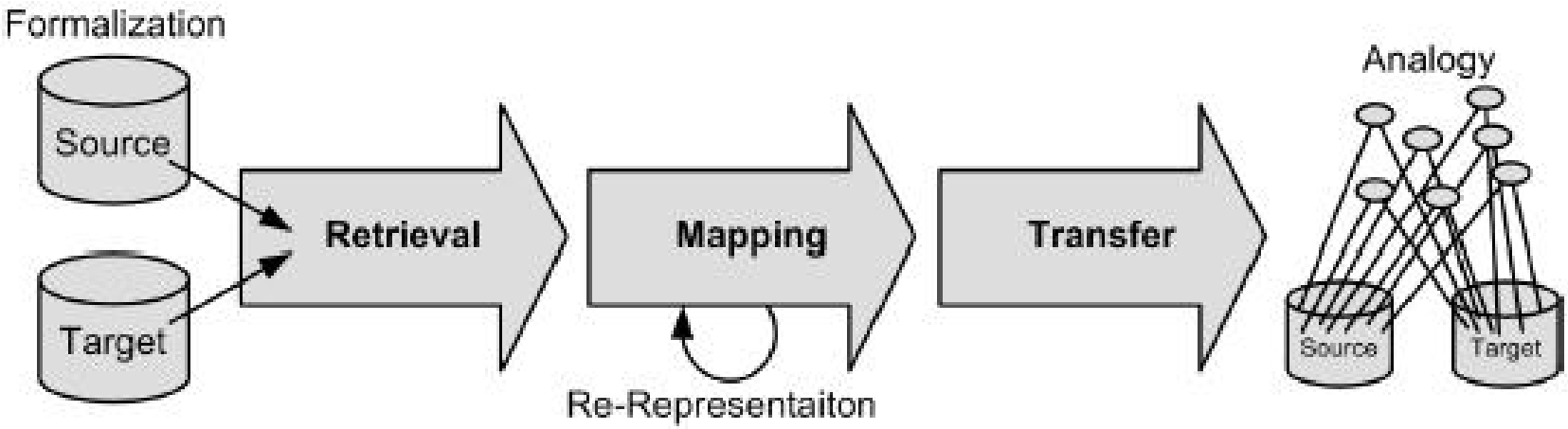
Analogy

Intermezzo: Analogy (1)

Analogy

- “ἀναλογία” - analogia, “proportion”.
- Informally: Claims of similarity, often used in argumentation or when explaining complex situations.
- A bit more formal: **Analogy-making is the human ability of perceiving dissimilar domains as similar with respect to certain aspects** based on shared commonalities in relational structure or appearance.
  (Incidental remark: In less complex forms also to be found in some other primates.)

Intermezzo: Analogy (2)



Non-Classical and Cross-Domain Reasoning

Heuristic-Driven Theory Projection (1)

Heuristic-Driven Theory Projection (HDTP)

- Computing **analogical relations and inferences** (domains given as many-sorted first-order logic representation/many-sorted term algebras) using a generalisation-based approach.
- Base and target of analogy defined in terms of **axiomatisations**, i.e., given by a finite set of formulae.
- Aligning pairs of formulae by means of **anti-unification** (extending classical Plotkin-style first-order anti-unification to a restricted form of higher-order anti-unification).
- Proof-of-concept applications in modelling mathematical reasoning and concept blending in mathematics.

Heuristic-Driven Theory Projection (2)

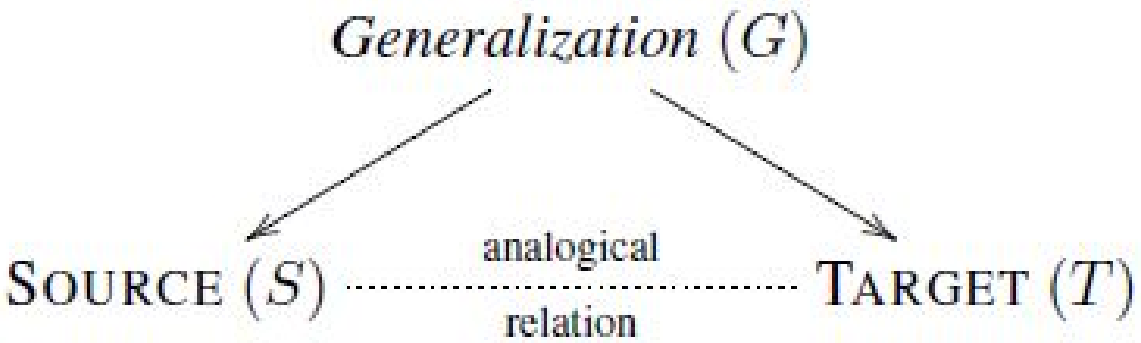


Figure: Analogy-making in HDTP.

Heuristic-Driven Theory Projection (3)

Anti-Unification

- Dual to the unification problem (see, e.g., logic programming or automated theorem proving).
- **Generalising terms** in a meaningful way, yielding for each term an **anti-instance** (distinct subterms replaced by variables).
- Goal: Finding the **most specific anti-unifier**.
- Plotkin: For a proper definition of generalisation, for a given pair of terms there always is exactly one least general generalisation (up to renaming of variables).
- Problem: Structural commonalities embedded in different contexts possibly not accessible by first-order anti-unification.

Heuristic-Driven Theory Projection (4)

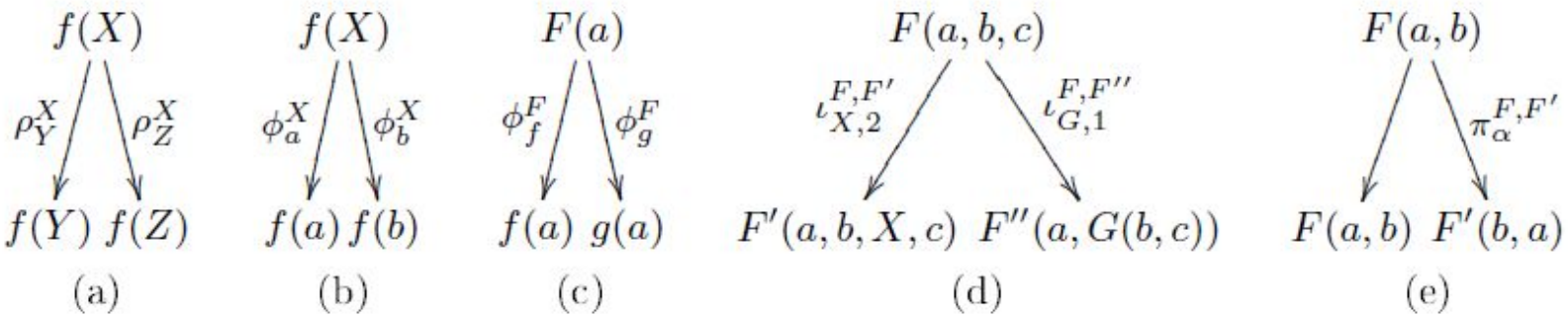
Restricted Higher-Order Anti-Unification

- First-order terms extended by **introducing variables taking arguments** (first-order variables become variables with arity 0), making a term either a first-order or a higher-order term.
- Class of **substitutions restricted** to (compositions of) the following four cases:
  1. Renamings $\rho^{F,F^*}\colon F(t_1,\ldots,t_n) \xrightarrow{\rho^{F,F^*}} F^*(t_1,\ldots,t_n)$.
  2. Fixations $\phi^F_c\colon F(t_1,\ldots,t_n) \xrightarrow{\phi^F_f} f(t_1,\ldots,t_n)$.
  3. Argument insertions $\iota^{F,F^*}_{G,i}$:

     $F(t_1,\ldots,t_n) \xrightarrow{\iota^{F,F^*}_{G,i}} F^*(t_1,\ldots,t_i,G(t_{i+1},\ldots,t_{i+k}),t_{i+k+1},\ldots,t_n)$.
  4. Permutations $\pi^{F,F^*}_\alpha\colon F(t_1,\ldots,t_n) \xrightarrow{\pi^{F,F^*}_\alpha} F^*(t_{\alpha(1)},\ldots,t_{\alpha(n)})$.

Heuristic-Driven Theory Projection (5)

Examples of higher-order anti-unifications:

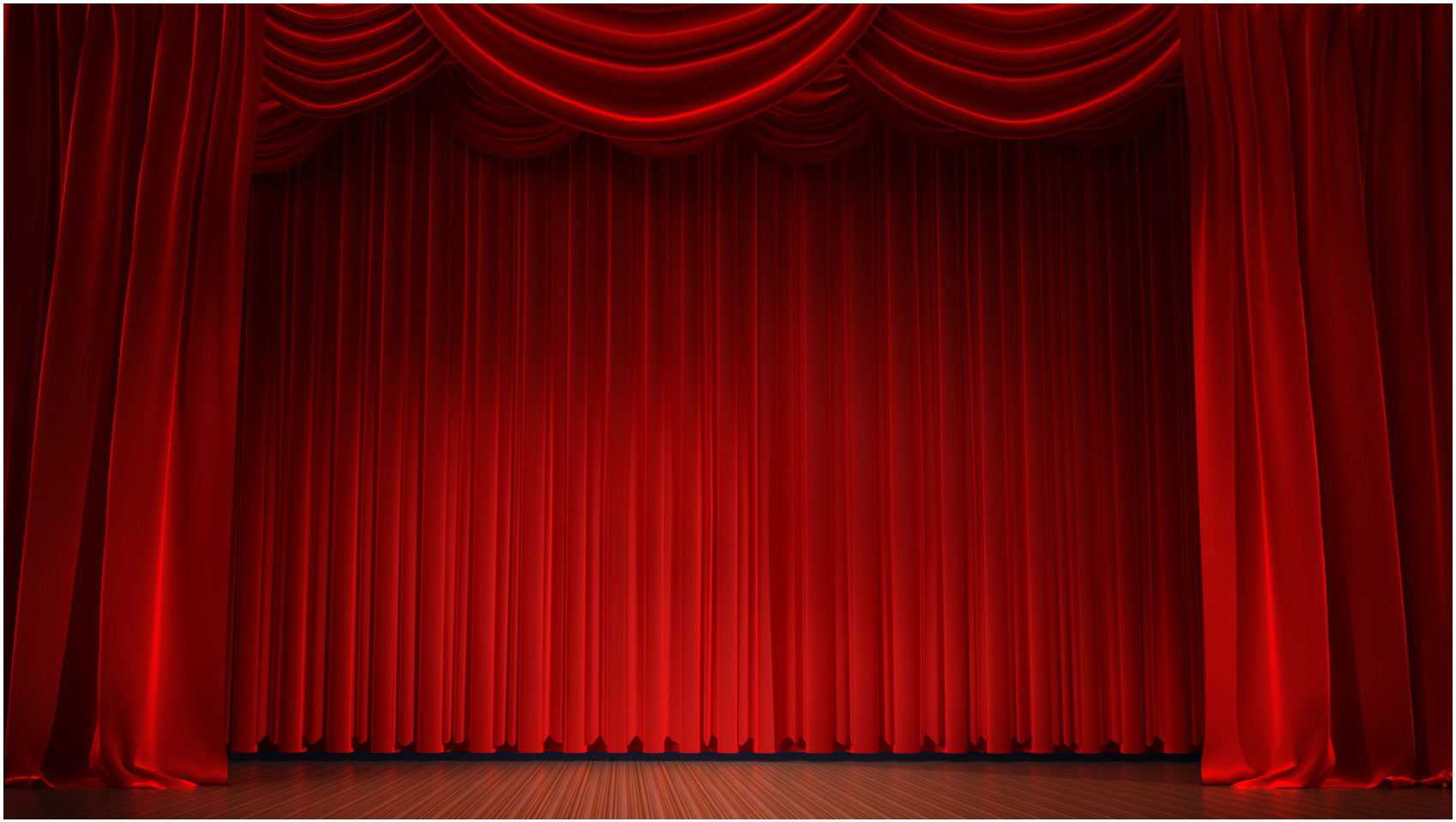


Heuristic-Driven Theory Projection (6)

Solar System	Rutherford Atom
sorts $real,\ object,\ time$ **entities** $sun : object,\ planet : object$ **functions** $mass : object \rightarrow real \times \{kg\}$ $dist : object \times object \times time \rightarrow real \times \{m\}$ $force : object \times object \times time \rightarrow real \times \{N\}$ $gravity : object \times object \times time \rightarrow real \times \{N\}$ $centrifugal : object \times object \times time \rightarrow real \times \{N\}$ **predicates** $revolves\_around : object \times object$ **facts** $\alpha_1 : mass(sun) > mass(planet)$ $\alpha_2 : mass(planet) > 0$ $\alpha_3 : \forall t : time : gravity(planet, sun, t) > 0$ $\alpha_4 : \forall t : time : dist(planet, sun, t) > 0$ **laws** $\alpha_5 : \forall t : time,\ o_1 : object,\ o_2 : object :$ $dist(o_1, o_2, t) > 0 \wedge gravity(o_1, o_2, t) > 0$ $\rightarrow centrifugal(o_1, o_2, t) = -gravity(o_1, o_2, t)$ $\alpha_6 : \forall t : time,\ o_1 : object,\ o_2 : object :$ $0 < mass(o_1) < mass(o_2) \wedge dist(o_1, o_2, t) > 0 \wedge$ $centrifugal(o_1, o_2, t) < 0$ $\rightarrow revolves\_around(o_1, o_2)$	**sorts** $real,\ object,\ time$ **entities** $nucleus : object,\ electron : object$ **functions** $mass : object \rightarrow real \times \{kg\}$ $dist : object \times object \times time \rightarrow real \times \{m\}$ $coulomb : object \times object \times time \rightarrow real \times \{N\}$ **facts** $\beta_1 : mass(nucleus) > mass(electron)$ $\beta_2 : mass(electron) > 0$ $\beta_3 : \forall t : time : coulomb(electron, nucleus, t) > 0$ $\beta_4 : \forall t : time : dist(electron, nucleus, t) > 0$

Heuristic-Driven Theory Projection (7)

types
$real,\ object,\ time$
constants
$X : object,\ Y : object$
functions
$mass : object \rightarrow real \times \{kg\}$
$dist : object \times object \times time \rightarrow real \times \{m\}$
$F : object \times object \times time \rightarrow real \times \{N\}$
$centrifugal : object \times object \times time \rightarrow real \times \{N\}$
predicates
$revolves\_around : object \times object \times object$
facts
$\gamma_1 : mass(X) > mass(Y)$
$\gamma_2 : mass(Y) > 0$
$\gamma_3 : \forall t : time : F(X, Y, t) > 0$
$\gamma_4 : \forall t : time : dist(X, Y, t) > 0$
laws
$\gamma_5* : \forall t : time,\ o_1 : object,\ o_2 : object :$
$dist(o_1, o_2, t) > 0 \wedge F(o_1, o_2, t) > 0$
$\rightarrow centrifugal(o_1, o_2, t) = -F(o_1, o_2, t)$
$\gamma_6* : \forall t : time,\ o_1 : object,\ o_2 : object :$
$0 < mass(o_1) < mass(o_2) \wedge dist(o_1, o_2, t) > 0 \wedge centrifugal(o_1, o_2, t) < 0$
$\rightarrow revolves\_around(o_1, o_2)$



Complexity and Tractability in Cognitive Models and Systems

Computer Metaphor and Church-Turing Thesis

Famous ideas at the heart of many endeavours in computational cognitive modelling and/or AI:

1. "Computer metaphor" of the mind (i.e. the concept of a computational theory of mind).
2. Church-Turing thesis.

1. Bridges gap between humans and computers:
   - Human mind and brain can be seen as information processing system.
   - Reasoning and thinking corresponds to computation as formal symbol manipulation.
2. Gives account of the nature and limitations of the computational power of such a system.

Computer Metaphor and Church-Turing Thesis

Famous ideas at the heart of many endeavours in computational cognitive modelling and/or AI:

1. "Computer metaphor" of the mind (i.e. the concept of a computational theory of mind).
2. Church-Turing thesis.

1. Bridges gap between humans and computers:
   - Human mind and brain can be seen as information processing system.
   - Reasoning and thinking corresponds to computation as formal symbol manipulation.
2. Gives account of the nature and limitations of the computational power of such a system.

P-Cognition Thesis

Significant impact on cognitive science and cognitive psychology:

- Explain human cognitive capacities modelled in terms of computational-level theories (i.e., as precise characterisations of hypothesised inputs and outputs of respective capacities together with functional mappings between them).
- **Problem:** Computational-level theories often underconstrained by available empirical data!

$\Rightarrow$ Use mathematical complexity theory as assisting tool:

NP-completeness!

P-Cognition thesis

Human cognitive capacities hypothesised to be of the polynomial-time computable type.

(*Interpretation: "Humans can comfortably solve non-trivial instances of this problem, where the exact size depends on the problem at hand".*)

P-Cognition Thesis

Significant impact on cognitive science and cognitive psychology:

- Explain human cognitive capacities modelled in terms of computational-level theories (i.e., as precise characterisations of hypothesised inputs and outputs of respective capacities together with functional mappings between them).
- **Problem:** Computational-level theories often underconstrained by available empirical data!

$\Rightarrow$ Use mathematical complexity theory as assisting tool:

NP-completeness!

P-Cognition thesis

Human cognitive capacities hypothesised to be of the polynomial-time computable type.

(*Interpretation: "Humans can comfortably solve non-trivial instances of this problem, where the exact size depends on the problem at hand".*)

“polynomial-time computable” = “efficient”?

- Humans able to solve problems which may be hard in general but feasible if certain parameters of the problem restricted.
- **Parametrised complexity theory**: “tractability” captured by FPT.[1]

FPT-Cognition thesis (van Rooij, 2008)

Human cognitive capacities hypothesised to be fixed-parameter tractable for one or more input parameters that are small in practice (i.e., computational-level theories have to be in FPT).

Tractable AGI thesis (Besold & Robere, 2013)

Models of cognitive capacities in artificial intelligence and computational cognitive systems have to be fixed-parameter tractable for one or more input parameters that are small in practice (i.e., have to be in FPT).

[1] A problem P is in FPT if P admits an $O(f(\kappa)n^c)$ algorithm, where n is the input size, κ is a parameter of the input constrained to be “small”, c is an independent constant, and f is some computable function.

“polynomial-time computable” = “efficient”?

- Humans able to solve problems which may be hard in general but feasible if certain parameters of the problem restricted.
- **Parametrised complexity theory**: “tractability” captured by FPT.[1]

FPT-Cognition thesis (van Rooij, 2008)

Human cognitive capacities hypothesised to be fixed-parameter tractable for one or more input parameters that are small in practice (i.e., computational-level theories have to be in FPT).

Tractable AGI thesis (Besold & Robere, 2013)
Models of cognitive capacities in artificial intelligence and computational cognitive systems have to be fixed-parameter tractable for one or more input parameters that are small in practice (i.e., have to be in FPT).

[1] A problem P is in FPT if P admits an $O(f(\kappa)n^c)$ algorithm, where n is the input size, κ is a parameter of the input constrained to be “small”, c is an independent constant, and f is some computable function.

“polynomial-time computable” = “efficient”?

- Humans able to solve problems which may be hard in general but feasible if certain parameters of the problem restricted.
- **Parametrised complexity theory**: “tractability” captured by FPT.[1]

FPT-Cognition thesis (van Rooij, 2008)

Human cognitive capacities hypothesised to be fixed-parameter tractable for one or more input parameters that are small in practice (i.e., computational-level theories have to be in FPT).

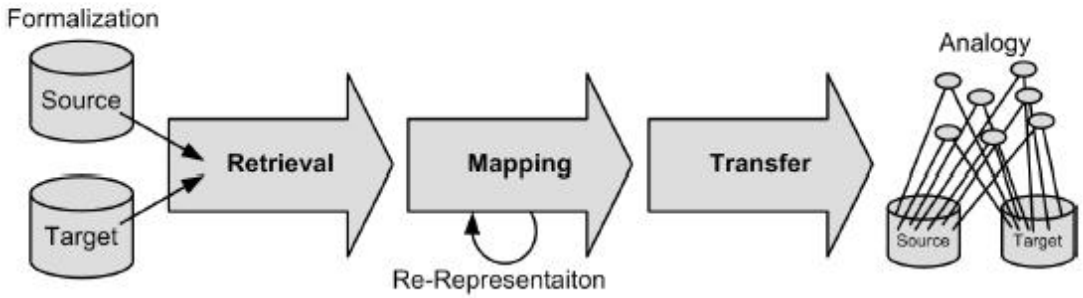
Tractable AGI thesis (Besold & Robere, 2013)
Models of cognitive capacities in artificial intelligence and computational cognitive systems have to be fixed-parameter tractable for one or more input parameters that are small in practice (i.e., have to be in FPT).

[1] A problem P is in FPT if P admits an $O(f(\kappa)n^c)$ algorithm, where n is the input size, κ is a parameter of the input constrained to be “small”, c is an independent constant, and f is some computable function.

Complexity of HDTP (1)

HDTP is naturally split into two mechanisms:

- Analogical matching of input theories.
- Re-representation of input theories by deduction in FOL.



$\Rightarrow$ Re-representation is undecidable (undecidability of FOL).

$\Rightarrow$ Focus on mechanism for analogical matching.

Complexity of HDTP (2)

Problem 1. F Anti-Unification
Input: Two terms f, g, and a natural $k \in \mathbb{N}$
Problem: Is there an anti-unifier h, containing at least k variables, using only **renamings and fixations**?

Problem 2. FP Anti-Unification
Input: Two terms f, g, and naturals $l, m, p \in \mathbb{N}$.
Problem: Is there an anti-unifier h, containing at least l 0-ary variables and at least m higher arity variables, and two substitutions σ, τ using only renamings, fixations, and **at most** p **permutations** such that $h \xrightarrow{\sigma} f$ and $h \xrightarrow{\tau} g$?

Problem 3. FPA Anti-Unification
Input: Two terms f, g and naturals $l, m, p, a \in \mathbb{N}$.
Problem: Is there an anti-unifier h, containing at least l 0-ary variables, at least m higher arity variables, and two substitutions σ, τ using renamings, fixations, at most p permutations, and **at most** a **argument insertions** such that $h \xrightarrow{\sigma} f$ and $h \xrightarrow{\tau} g$?

Complexity of HDTP (3)

...a fair share of formal magic involving a Canadian and some "Subgraph Isomorphism to Clique" reductions later...

Complexity of HDTP (Higher-Order Anti-Unification)

1. **F Anti-Unification is solvable in polynomial time**.
2. Let m denote the minimum number of higher arity variables and let p be the maximum number of permutations applied. Then **FP Anti-Unification is** NP**-complete and** W[1]**-hard w.r.t. parameter set** $\{m, p\}$.
3. Let r be the maximum arity and s be the maximum number of subterms of the input terms. Then **FP Anti-Unification is in** FPT **w.r.t. parameter set** $\{s, r, p\}$.
4. **FPA Anti-Unification is** NP**-complete and** W[1]**-hard w.r.t. parameter set** $\{m, p, a\}$.

(For proofs: R. Robere and T. R. Besold. ***Complex Analogies: Remarks on the Complexity of HDTP***. In Proceedings of the 25th Australasian Joint Conference on Artificial Intelligence (AI 2012), LNCS 7691. Springer, 2012.)

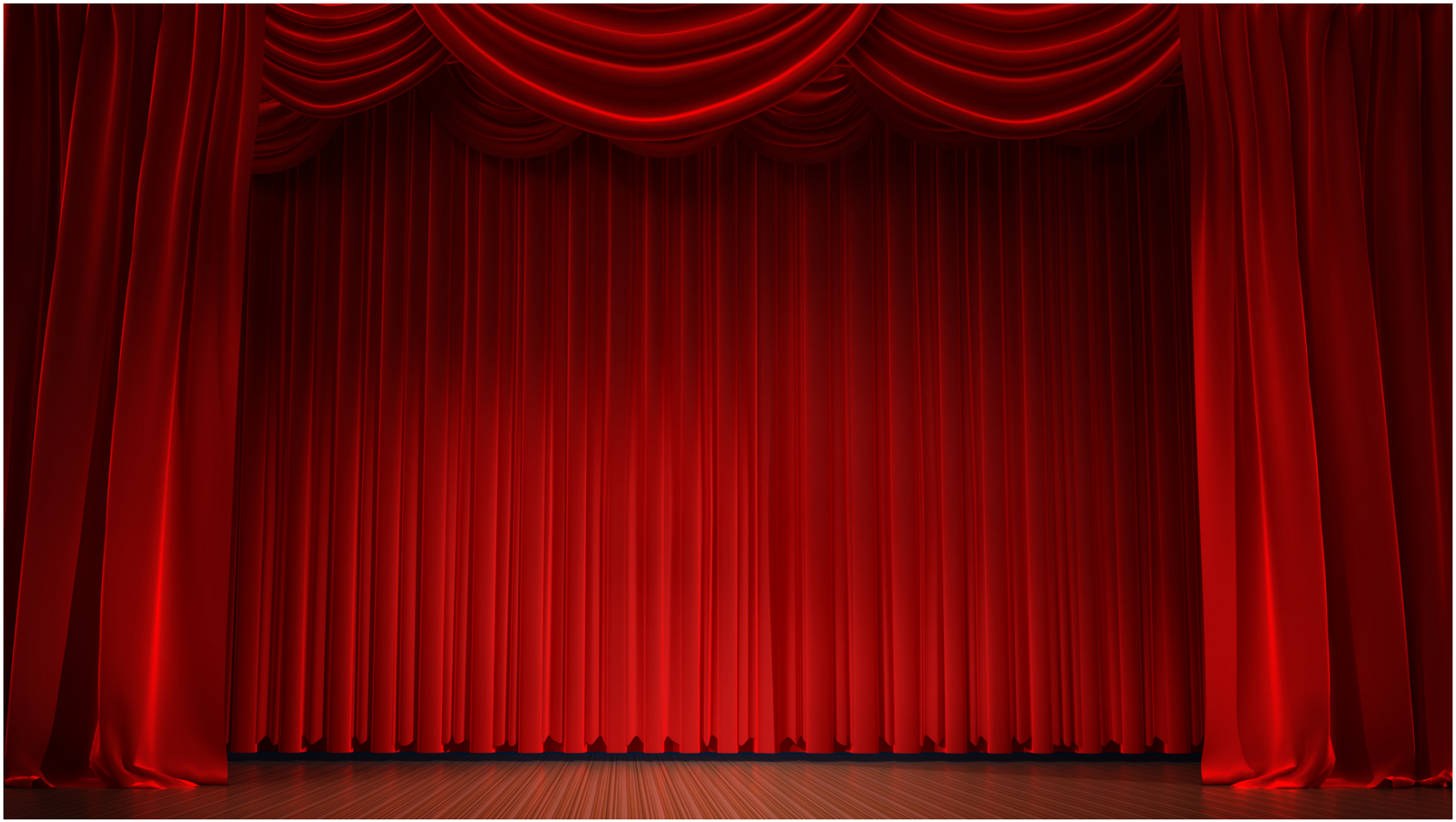
Complexity of HDTP (3)

...a fair share of formal magic involving a Canadian and some "Subgraph Isomorphism to Clique" reductions later...

Complexity of HDTP (Higher-Order Anti-Unification)

1. **F Anti-Unification is solvable in polynomial time**.
2. Let m denote the minimum number of higher arity variables and let p be the maximum number of permutations applied. Then **FP Anti-Unification is** NP**-complete and** W[1]**-hard w.r.t. parameter set** $\{m, p\}$.
3. Let r be the maximum arity and s be the maximum number of subterms of the input terms. Then **FP Anti-Unification is in** FPT **w.r.t. parameter set** $\{s, r, p\}$.
4. **FPA Anti-Unification is** NP**-complete and** W[1]**-hard w.r.t. parameter set** $\{m, p, a\}$.

(For proofs: R. Robere and T. R. Besold. ***Complex Analogies: Remarks on the Complexity of HDTP***. In Proceedings of the 25th Australasian Joint Conference on Artificial Intelligence (AI 2012), LNCS 7691. Springer, 2012.)



Concept Blending

Concept blending: A + B = ? (1)

Concept blending: A + B = ? (2)

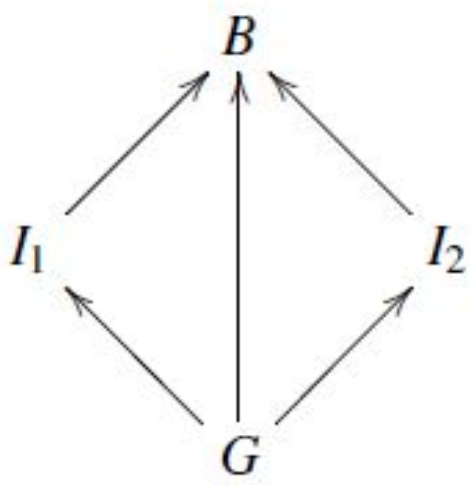
Concept blending: A + B = ? (3)

Concept blending: A + B = ? (4)

Foundations of Theory Blending (1)

Concept Blending

- Given two domain theories I_1 and I_2, representing two conceptualisations...
- ...look for a generalisation G...
- ...construct the blend space B in such a way as to preserve the correlations between I_1 and I_2 established by G.



Foundations of Theory Blending (2)

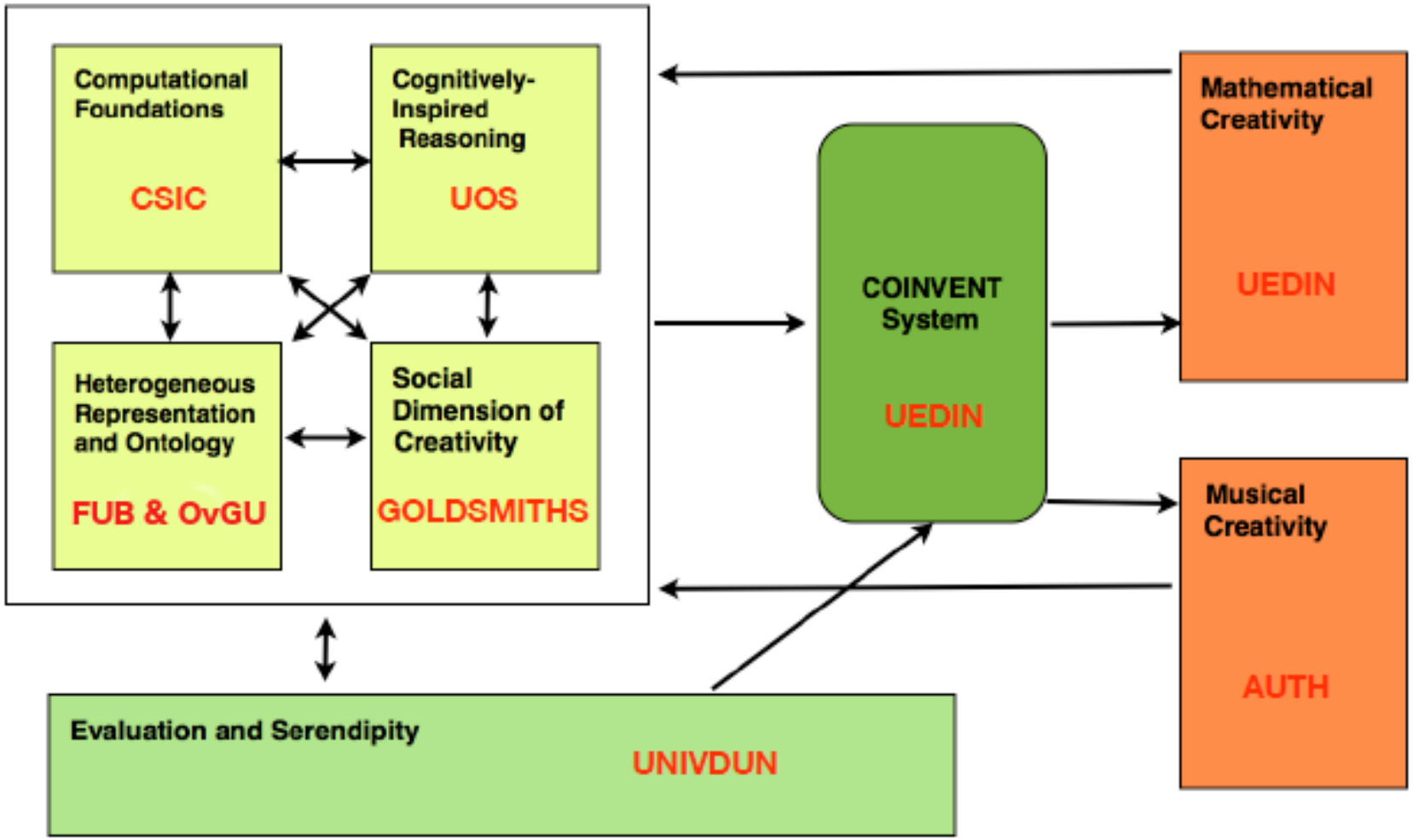
Example: Houseboat vs. boathouse

- Concept blends of *HOUSE* and *BOAT* into *BOATHOUSE* and *HOUSEBOAT*.
  - $I_1 = \{HOUSE \sqsubseteq \forall LIVES-IN.RESIDENT\}$
  - $I_2 = \{BOAT \sqsubseteq \forall RIDES-ON.PASSENGER\}$
- *HOUSEBOAT*: Aligning parts of the conceptual spaces...
  - $RESIDENT \leftrightarrow PASSENGER$
  - $LIVES-IN \leftrightarrow RIDES-ON$
  - $HOUSE \leftrightarrow BOAT$
- *BOATHOUSE*: Aligning parts of the conceptual spaces...
  - $RESIDENT \leftrightarrow BOAT$

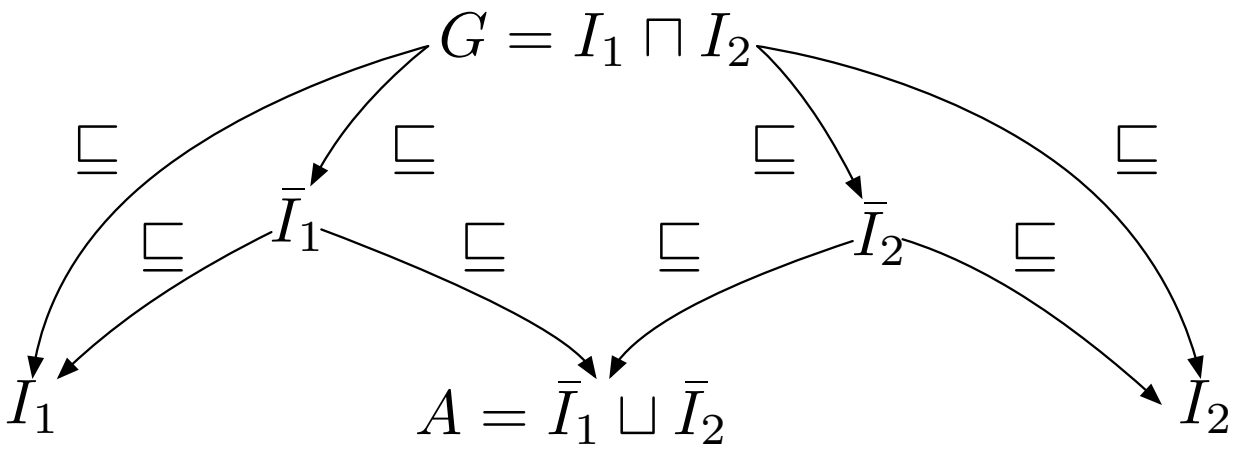
The Concept Invention Theory (COINVENT) Project (1)

- To develop a **novel, computationally feasible, formal model of conceptual blending** based on Fauconnier and Turner's theory.
- To gain **a deeper understanding of conceptual blending** and its role in computational creativity.
- To design **a generic, creative computational system** capable of serendipitous invention and manipulation of novel abstract concepts.
- To validate our model and its computational realisation in two representative working domains: **mathematics and music**.

The Concept Invention Theory (COINVENT) Project (2)



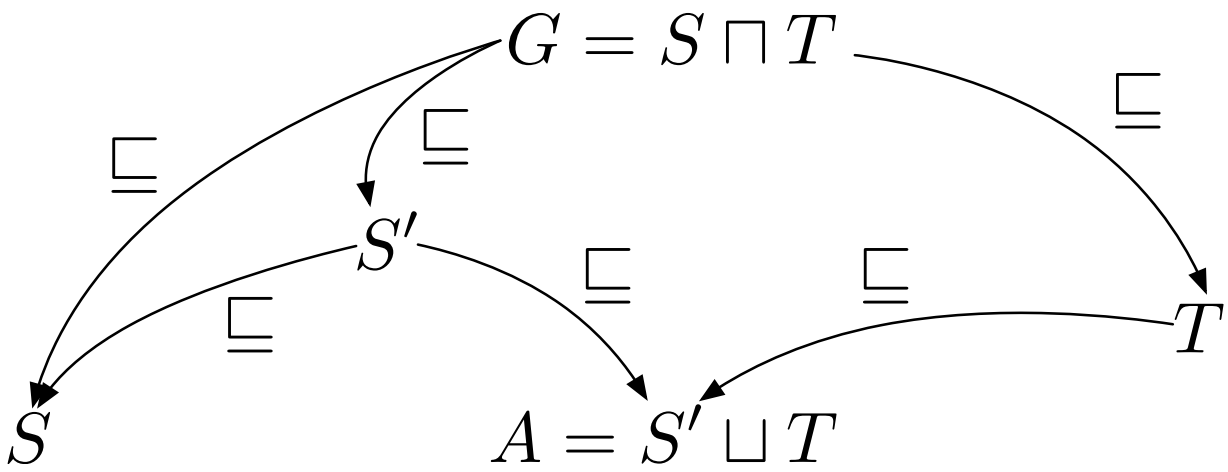
Amalgamation 101 (1)



Amalgam

A description $A \in \mathcal{L}$ is an amalgam of two inputs I_1 and I_2 (with anti-unification $G = I_1 \sqcap I_2$) if there exist two generalisations $\bar{I}_1$ and $\bar{I}_2$ such that (1) $G \sqsubseteq \bar{I}_1 \sqsubseteq I_1$, (2) $G \sqsubseteq \bar{I}_2 \sqsubseteq I_2$, and (3) $A = \bar{I}_1 \sqcup \bar{I}_2$

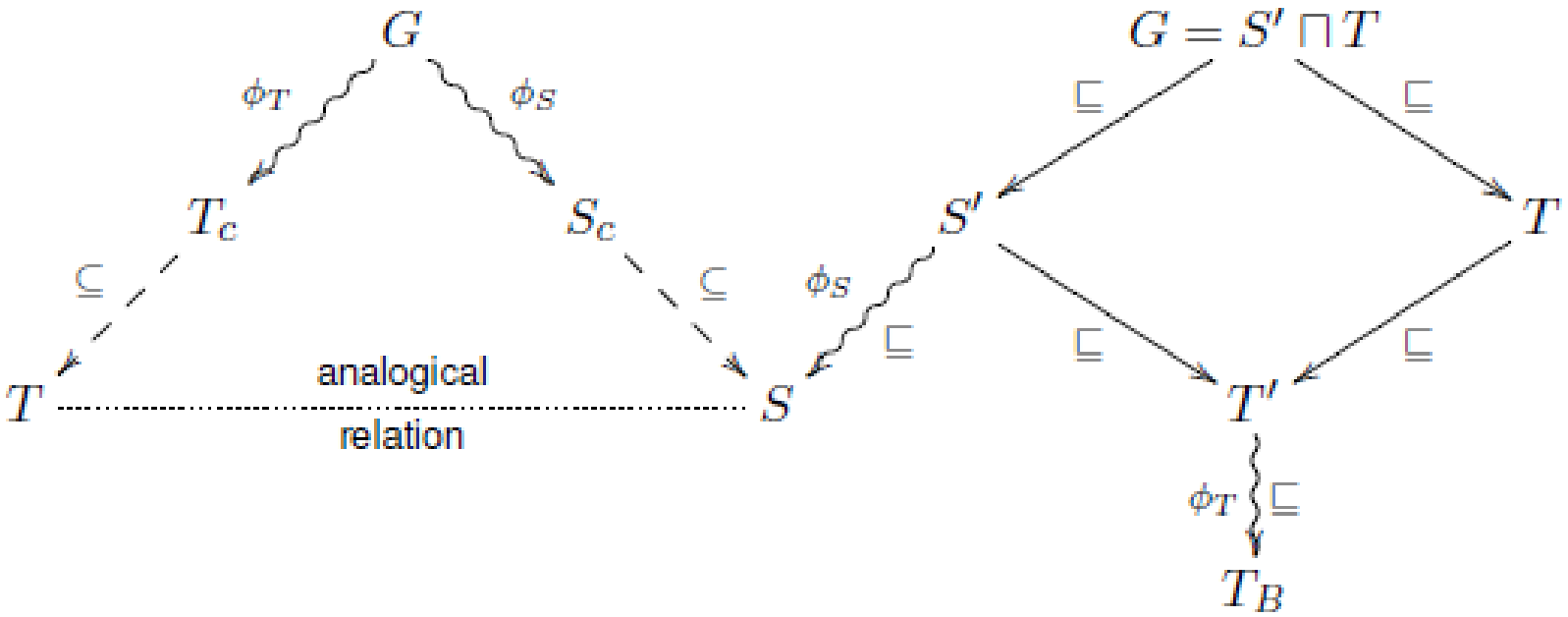
Amalgamation 101 (2)



Asymmetric Amalgam

An asymmetric amalgam $A \in \mathcal{L}$ of two inputs S (source) and T (target) satisfies that $A = S' \sqcup T$ for some generalisation of the source $S' \sqsubseteq S$.

COINVENT's Blending Schema



1.) Compute shared generalisation G from S and T with $\phi_S(G) = S_c$.

2.) Re-use ϕ_S in generalisation of S into S'.

3.) Combine S' in asymmetric amalgam with T into proto-blend $T' = S' \sqcup T$.

4.) By application of ϕ_T, complete T' into blended output theory T_B.

($\subseteq$: element-wise subset relationship between sets of axioms. $\sqsubseteq$: subsumption between theories in direction of respective arrows.)

...and the Implementation?

- Use **HDTP for computation of generalisation(s)** and substitution chains/higher-order anti-unifications.
- Currently: Restrict HDTP to **using only renamings and fixations**.
  $\Rightarrow$ Possibility to use "classical" **semantic consequence** $\models$ **as ordering relationship**.
  (Also preserved by later unifications and addition of axioms.)
- Use **HDTP's heuristics for selecting least general generalisation** *G* (among several options).
- Currently: Naive **consistency/inconsistency check with final blend** (both internally and against world knowledge).
  $\Rightarrow$ **Clash resolution by re-start with reduced set of input axioms.**

...and the Implementation?

- Use **HDTP for computation of generalisation(s)** and substitution chains/higher-order anti-unifications.
- Currently: Restrict HDTP to **using only renamings and fixations**.
  $\Rightarrow$ Possibility to use "classical" **semantic consequence** $\models$ **as ordering relationship**.
  (Also preserved by later unifications and addition of axioms.)
- Use **HDTP's heuristics for selecting least general generalisation** G (among several options).
- Currently: Naive **consistency/inconsistency check with final blend** (both internally and against world knowledge).
  $\Rightarrow$ **Clash resolution by re-start with reduced set of input axioms.**

Example: Brillo, the Foldable Toothbrush

Stereotypical characterization for a pocketknife:

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

Stereotypical characterization for a toothbrush:

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

Computing a shared generalization:

Applied substitutions:

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

Applied substitutions:

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, handle)

Applied substitutions:
pocketknife, toothbrush => E

has_part(pocketknife, handle)

has_part(toothbrush, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, handle)

Applied substitutions:
pocketknife, toothbrush => E

has_part(pocketknife, handle)

has_part(toothbrush, handle)

has_part(pocketknife, blade)

has_part(toothbrush, brush_head)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(toothbrush, brush)

has_functionality(pocketknife, fold)

has_part(E, handle)

Applied substitutions:
pocketknife, toothbrush => E

has_part(pocketknife, handle)

has_part(toothbrush, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, handle)

has_part(E, P)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, handle)

has_part(E, P)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, handle)

has_part(E, P)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, handle)

has_part(E, P)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

Shared generalization:

has_part(E, handle)

has_part(E, P)

has_functionality(E, F)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

Computing the generalized source theory:

has_part(E, handle)

has_part(E, P)

has_functionality(E, F)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

Computing the generalized source theory:

has_part(E, handle)

has_part(E, P)

has_functionality(E, F)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, handle)

has_part(E, P)

has_functionality(E, P)

has_part(E, hinge)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, handle)

has_part(E, P)

has_functionality(E, P)

has_part(E, hinge)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, handle)

has_part(E, P)

has_functionality(E, P)

has_part(E, hinge)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

Generalized source theory:

has_part(E, handle)

has_part(E, P)

has_functionality(E, P)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

Computing the proto-blend/ asymmetric amalgam:

has_part(E, handle)

has_part(E, P)

has_functionality(E, P)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, handle)

has_part(E, P)

has_functionality(E, P)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, handle)

has_part(E, P)

has_functionality(E, P)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, handle)

has_part(E, P)

has_functionality(E, P)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, handle)

has_part(E, P)

has_functionality(E, P)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, handle)

has_part(E, P)

has_functionality(E, P)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(E, P)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(E, P)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(E, P)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

Proto-blend/ asymmetric amalgam:

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

Computing the final blend:

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(E, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

Final blend:

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)

has_part(toothbrush, hinge)

has_functionality(E, fold)

Applied substitutions:
pocketknife, toothbrush => E
blade, brush_head => P
cut, brush => F

has_part(pocketknife, handle)

has_part(pocketknife, blade)

has_part(pocketknife, hinge)

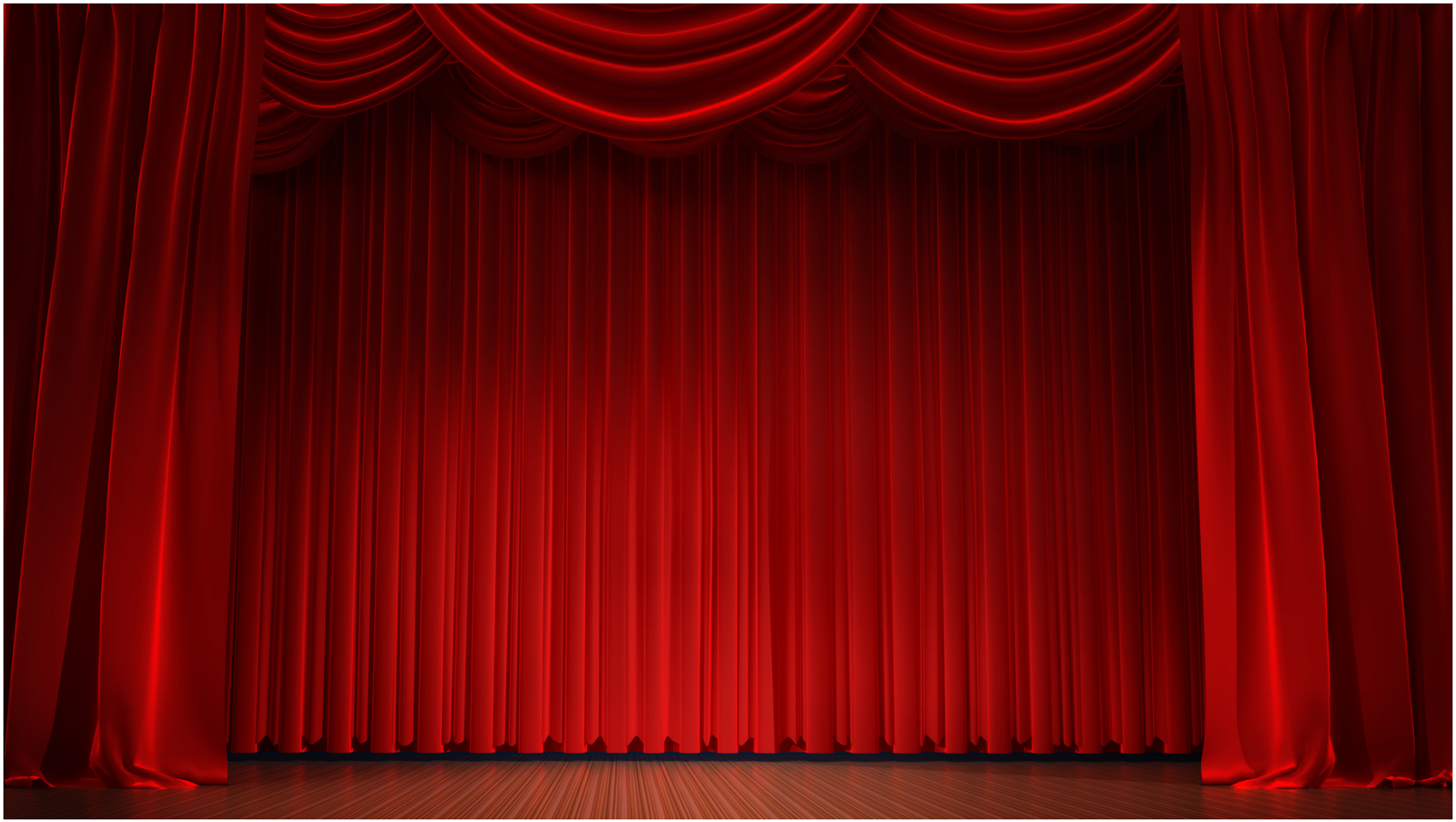
has_functionality(pocketknife, cut)

has_functionality(pocketknife, fold)

has_part(toothbrush, handle)

has_part(toothbrush, brush_head)

has_functionality(toothbrush, brush)



Conclusion

(Definitely Not) The End!

If you are interested in non-classical reasoning, tractability, approximability and similar topics in A(G)I and/or cognitive science, you are happily invited to...

1. ...talk to me after the presentation.
2. ...get in touch by e-mail:

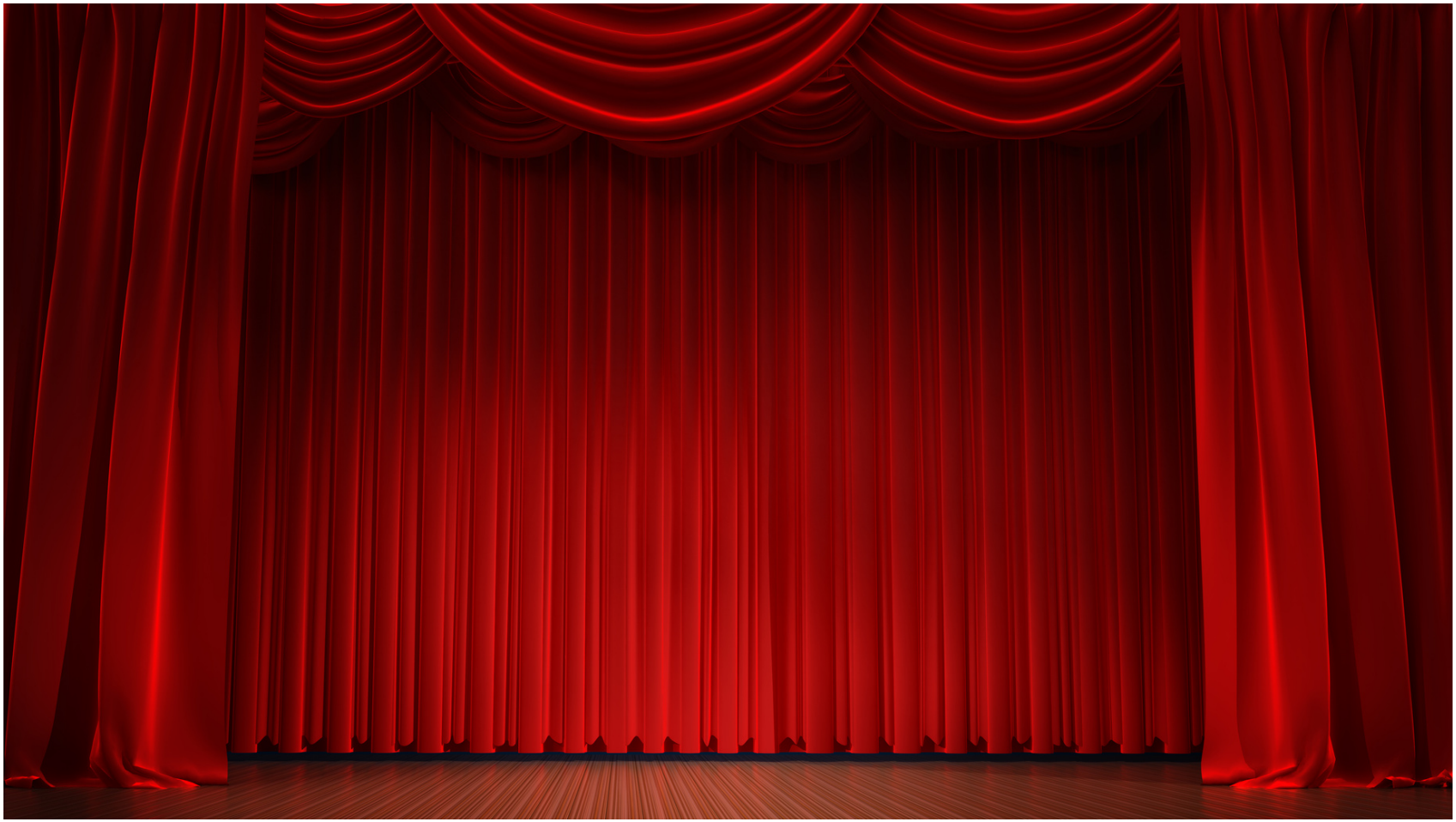
   TarekRichard.Besold@unibz.it.
3. ...occasionally have a look at our publications.[2]

---

[2] For instance:

Besold, T. R., and Robere, R.. *When Thinking Never Comes to a Halt: Using Formal Methods in Making Sure Your AI Gets the Job Done Good Enough*. In V. C. Müller (ed.), Fundamental Issues of Artificial Intelligence (Synthese Library, vol. 376). Springer, 2016.

Besold, T. R., and Plaza, E. *Generalize and Blend: Concept Blending Based on Generalization, Analogy, and Amalgams*. In H. Toivonen, S. Colton, M. Cook, and D. Ventura, Proceedings of the Sixth International Conference on Computational Creativity (ICCC) 2015. Brigham Young University Press, 2015.



Postludium

Disclaimer

- **Frequent criticism**:
  - Demanding for cognitive systems and models to work within certain complexity limits overly restrictive.
  - Maybe: Human mental activities actually performed as exponential-time procedures, but never noticed as exponent for some reason always very small.
- **Reply**:
  - Possibility currently cannot be excluded.
  - Instead: No claim that cognitive processes without exception within FPT, APX, FPA, or what-have-you...
  - ...but staying within boundaries makes cognitive systems and models plausible candidates for application in resource-bounded general-purpose cognitive agents.

Disclaimer

- **Frequent criticism**:
  - Demanding for cognitive systems and models to work within certain complexity limits overly restrictive.
  - Maybe: Human mental activities actually performed as exponential-time procedures, but never noticed as exponent for some reason always very small.
- **Reply**:
  - Possibility currently cannot be excluded.
  - Instead: No claim that cognitive processes without exception within FPT, APX, FPA, or what-have-you...
  - ...but staying within boundaries makes cognitive systems and models plausible candidates for application in resource-bounded general-purpose cognitive agents.

Complexity of the First HDTP Generation

As an aside:
Once upon a time, there was HDTP-old based on reducing certain higher-order to first-order anti-unifications by introduction of subterms built from "admissible sequences" over equational theories (i.e., conjunctions of FOL formulae with equality over a term algebra).

Complexity of HDTP-old

1. **HDTP-old is NP-complete.**
2. **HDTP-old is W[2]-hard with respect to a minimal bound on the cardinality of the set of all subterms of the term against which admissibility is checked.**

(For proofs: R. Robere and T. R. Besold. ***Complex Analogies: Remarks on the Complexity of HDTP***. In Proceedings of the 25th Australasian Joint Conference on Artificial Intelligence (AI 2012), LNCS 7691. Springer, 2012.)

Complexity of the First HDTP Generation

As an aside:
Once upon a time, there was HDTP-old based on reducing certain higher-order to first-order anti-unifications by introduction of subterms built from "admissible sequences" over equational theories (i.e., conjunctions of FOL formulae with equality over a term algebra).

Complexity of HDTP-old

1. **HDTP-old is** NP**-complete**.
2. **HDTP-old is** W[2]**-hard with respect to a minimal bound on the cardinality of the set of all subterms of the term against which admissibility is checked**.

(For proofs: R. Robere and T. R. Besold. ***Complex Analogies: Remarks on the Complexity of HDTP***. In Proceedings of the 25th Australasian Joint Conference on Artificial Intelligence (AI 2012), LNCS 7691. Springer, 2012.)

Approximability Classes

In the following, let...

- ...PTAS denote the class of all NP optimisation problems that admit a polynomial-time approximation scheme.
- ...APX be the class of NP optimisation problems allowing for constant-factor approximation algorithms.
- ...APX-*poly* be the class of NP optimisation problems allowing for polynomial-factor approximation algorithms.

Please note that PTAS $\subseteq$ APX $\subseteq$ APX-*poly* (with each inclusion being proper in case P $\neq$ NP).

Approximability Analysis of HDTP (1)

- FP Anti-Unification W[1]-hard to compute for parameter set m, p (m number of higher-arity variables, p number of permutations).

  $\Rightarrow$ No polynomial-time algorithm computing "sufficiently complex" generalisations (i.e., with lower bound on number of higher-arity variables), upper bounding number of permutations (W[1]-hardness for single permutation).

- **What if one considers generalisations which merely approximate the "optimal" generalisation in some sense?**

Approximability Analysis of HDTP (2)

Complexity of a Substitution

The complexity of a basic substitution σ is defined as

$$C(s) = \begin{cases} 0, & \text{if } \sigma \text{ is a renaming.} \\ 1, & \text{if } \sigma \text{ is a fixation or permutation.} \\ k+1, & \text{if } \sigma \text{ is a } k\text{-ary argument insertion.} \end{cases}$$

The complexity of a restricted substitution $\sigma = \sigma_1 \circ \cdots \circ \sigma_n$ (i.e., the composition of any sequence of unit substitutions) is the sum of the composed substitutions: $C(\sigma) = \sum_{i=1}^{n} C(\sigma_i)$.

Approximability Analysis of HDTP (3)

Consider problem of finding generalisation which **maximises complexity over all generalisations**:

- Complex generalisation would contain "most information" present over all of the generalisations chosen (i.e., maximising the "information load").
- Using approximability results on MAXCLIQUE:

Approximation Complexity of HDTP Analogy-Making

FP anti-unification is not in APX (i.e., does not allow for constant-factor approximation algorithms) and is hard for APX-poly.

(For proofs: T. R. Besold and R. Robere. ***When Almost Is Not Even Close: Remarks on the Approximability of HDTP***. In Artificial General Intelligence - 6th International Conference (AGI 2013), LNCS. Springer, 2013.)