July 7th, 2015

ASP, Amalgamation, and the Conceptual Blending Workflow

Manfred Eppe[1,5], Ewen Maclean[2], Oliver Kutz[3], Roberto Confalonieri[1], Marco Schorlemmer[1], Enric Plaza[1]

[1]IIIA-CSIC, Barcelona, Spain – [2]University of Edinburgh, UK – [3]University of Bozen-Bolzano, Italy – [4]University of Magdeburg, Germany – [5]International Computer Science Institute, Berkeley, USA

Conceptual Blending

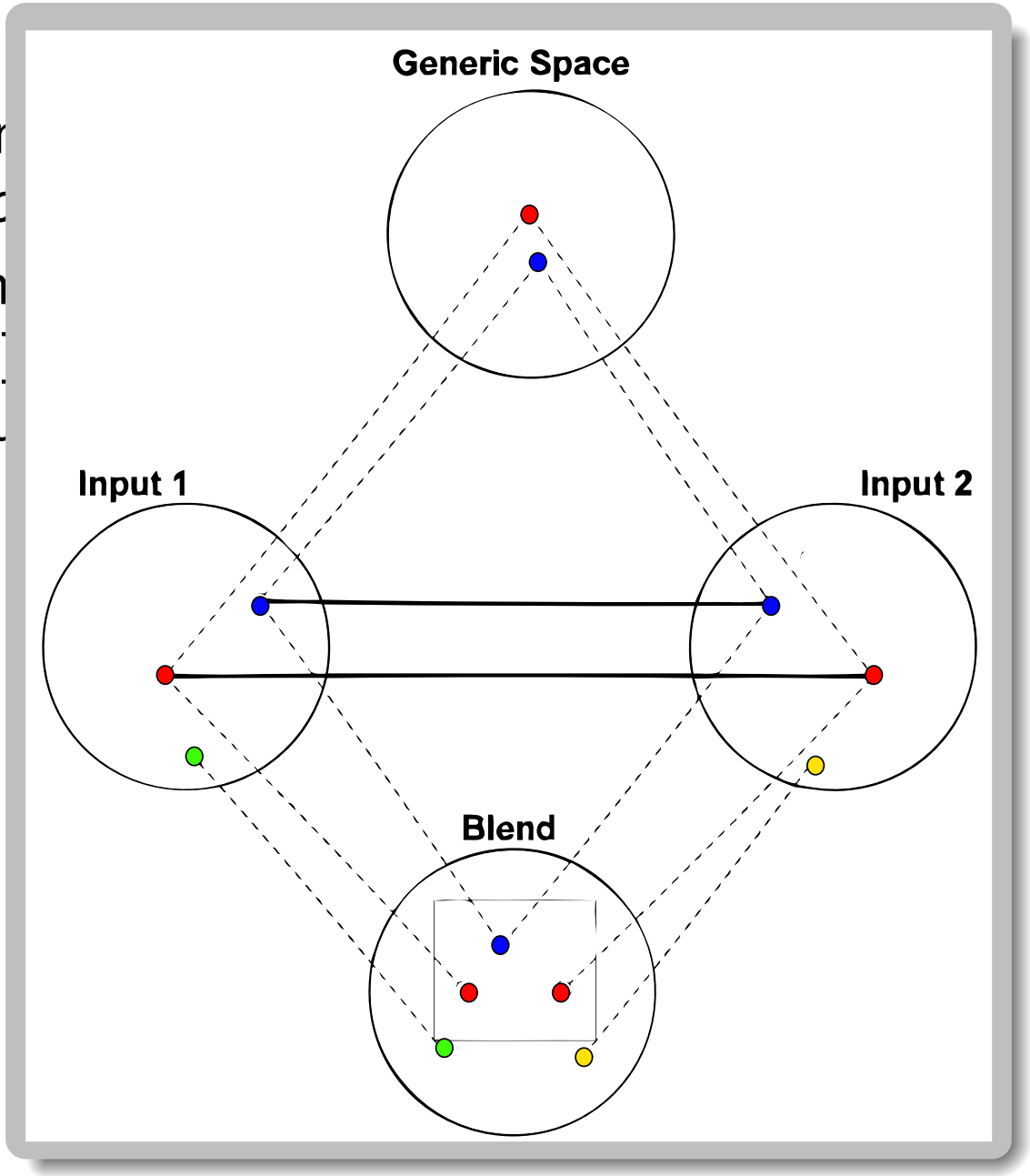
- Unfamiliar combination of familiar ideas [Boden, 1996] – *Bisociation* by Koestler [1964].

Conceptual Blending

- Unfamiliar combination of familiar ideas [Boden, 1996] – *Bisociation* by Koestler [1964].
- Cognitive theory described by Fauconnier and Turner [1998, 2002], Turner [2014].
  The universal creative engine of human thinking.

Conceptual Blending

- Unfar 1996] –
  Bisoc
- Cogn urner [1998,
  2002]
  The u .



Conceptual Blending

- Unfamiliar combination of familiar ideas [Boden, 1996] – *Bisociation* by Koestler [1964].
- Cognitive theory described by Fauconnier and Turner [1998, 2002], Turner [2014].
  The universal creative engine of human thinking.
- Hypothetic explanation for the '**human spark**' – the beginning of rapid cultural development approx. 32,000 years ago.

(The lion man)

Conceptual Blending

- Unfamiliar combination of familiar ideas [Boden, 1996] – *Bisociation* by Koestler [1964].
- Cognitive theory described by Fauconnier and Turner [1998, 2002], Turner [2014].
  The universal creative engine of human thinking.
- Hypothetic explanation for the '**human spark**' – the beginning of rapid cultural development approx. 32,000 years ago.
- Our demo domains:
  mathematics and music

(The lion man)

Example – Representing Mathematical Theories and Inventing Eureka Lemmas

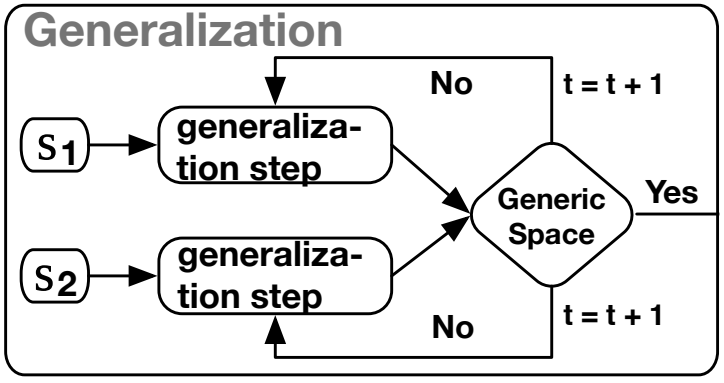
spec NAT $=$
sorts Nat
ops $zero : Nat$;
$s : Nat \to Nat$
$plus : Nat \times Nat \to Nat$
$sum : Nat \to Nat$
$qsum : Nat \times Nat \to Nat$
$\forall\ x, y : Nat$
%% Target theorem to prove:
(NT) $sum(x) = qsum(x, zero)$
%% Creative eureka lemma:
(NL) $plus(sum(x), y) = qsum(x, y)$
end

Example – Representing Mathematical Theories and Inventing Eureka Lemmas

spec NAT =
sorts Nat
ops $zero : Nat$;
$s : Nat \to Nat$
$plus : Nat \times Nat \to Nat$
$sum : Nat \to Nat$
$qsum : Nat \times Nat \to Nat$
$\forall\ x, y : Nat$
%% Target theorem to prove:
(NT) $sum(x) = qsum(x, zero)$
%% Creative eureka lemma:
(NL) $plus(sum(x), y) = qsum(x, y)$
end

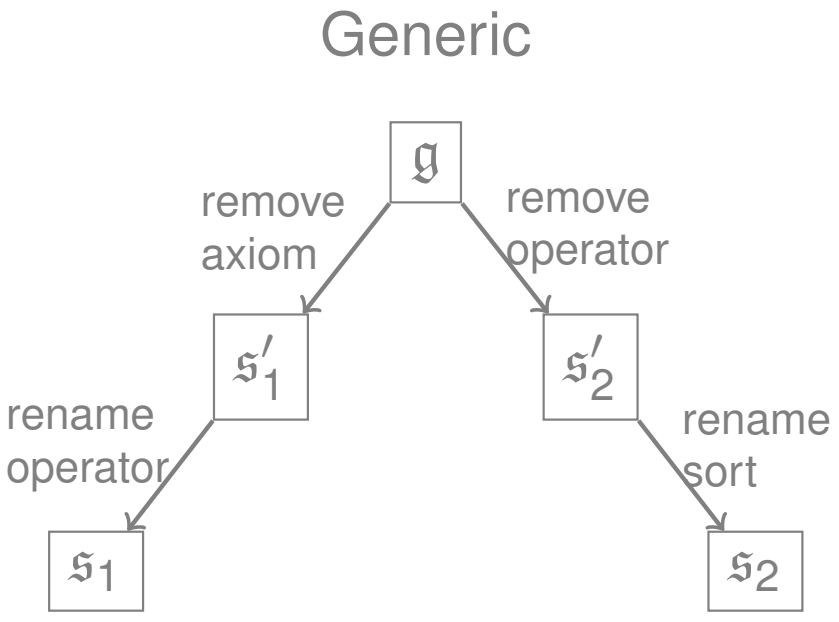
spec LIST =
sorts El
L
ops $nil : L$;
$cons : El \times L \to L$;
$app : L \times L \to L$;
$rev : L \to L$;
$qrev : L \times L \to L$
$\forall\ x, y : L; h : El$
%% Target theorem to prove:
(LT) $rev(x) = qrev(x, nil)$
%% Creative eureka lemma:
(LL) $app(rev(x), y) = qrev(x, y)$
end

Amalgamation Workflow (I)



Generalization

1. Find *generic space* in step-wise search process implemented in ASP
2. Output upside-down 'V' graphs with generalized versions of input spaces



Generalizing specifications as planning problem in ASP

- Elements as parts of a specification:

$$hasOp(\mathfrak{s}, o, t)$$
$$opHasArgSort(\mathfrak{s}, o, s, pos, t)$$
$$\vdots$$

- Generalization operations for removal and renaming with preconditions and effects, e.g., rename operator:

$$\begin{aligned} poss(renameOp(o_1, o_2, \mathfrak{s}_2), \mathfrak{s}_1, t) \leftarrow\ & argSortsEquivalent(o_1, o_2, \mathfrak{s}_1, \mathfrak{s}_2, t), \\ & rangeSortsEquivalent(o_1, o_2, \mathfrak{s}_1, \mathfrak{s}_2, t), \\ hasOp(\mathfrak{s}, o_2, t+1) \leftarrow\ & exec(renameOp(o_1, o_2, \mathfrak{s}_2), \mathfrak{s}_1, t). \\ hasOp(\mathfrak{s}, o_1, t+1) \leftarrow\ & \text{“}o_1 \text{ has not been removed or renamed at } t\text{”} \end{aligned}$$

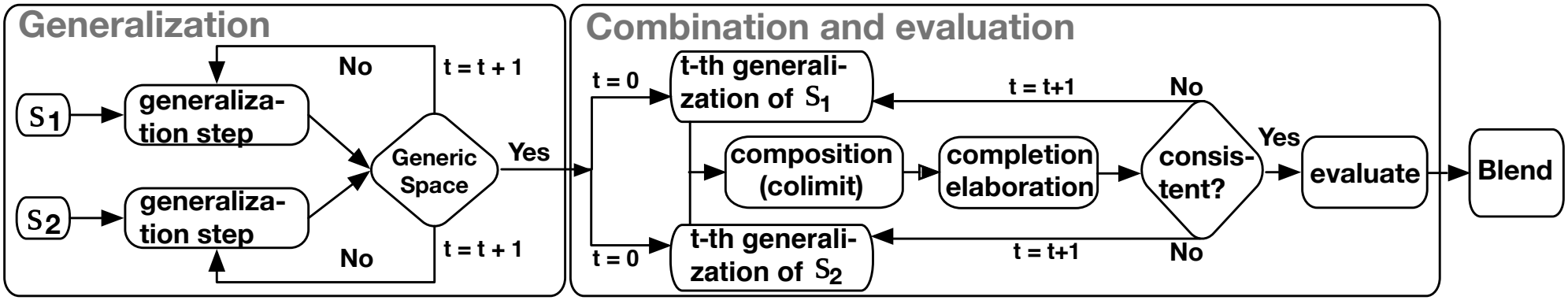
- Choice rule to span up search space:

$$0\{exec(A, \mathfrak{s}, t) : poss(A, \mathfrak{s}, t)\}1 \leftarrow spec(\mathfrak{s}).$$

- Integrity constraint to assure that generic space has been found (operators, sorts and axioms are equivalent):

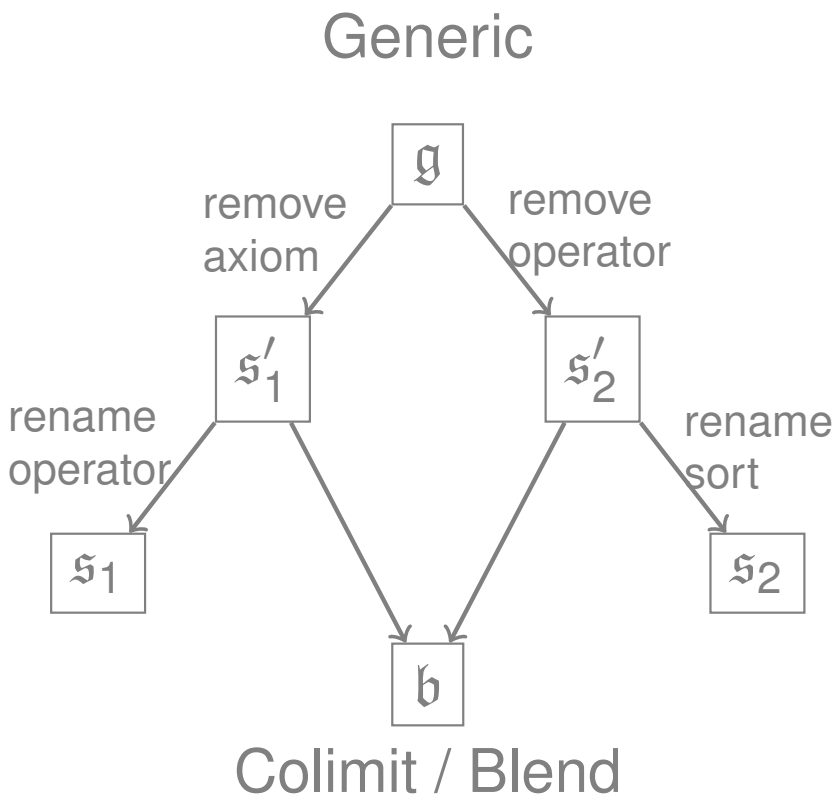
$$\leftarrow notEquivalent(\mathfrak{s}_1, \mathfrak{s}_2, t)$$

Amalgamation Workflow (II)

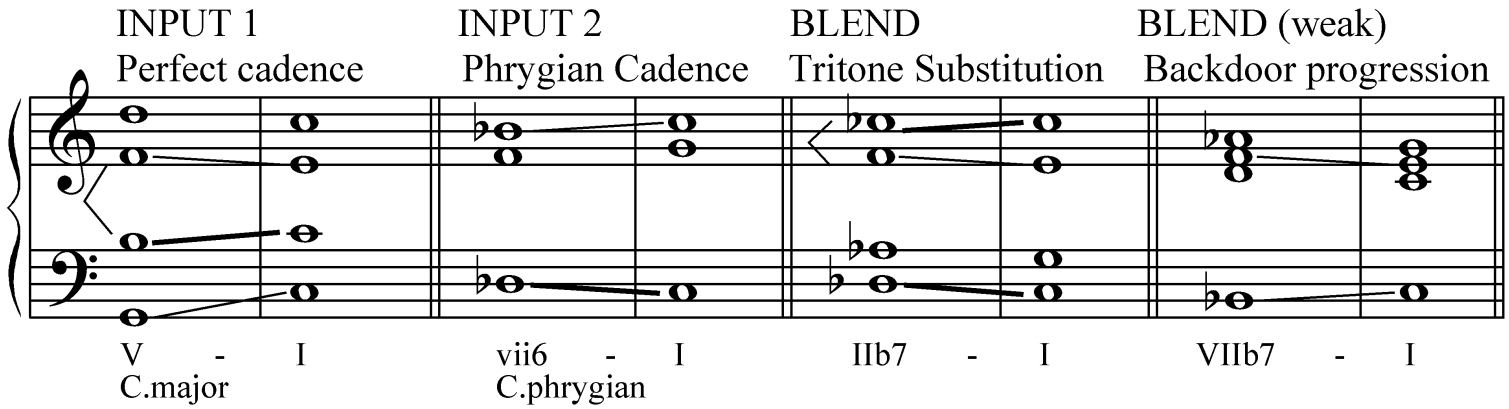


Combination and evaluation

1. Compose pairs of generalizations (colimit)
2. Complete and elaborate composition (deduction, other external tools)
3. If the blend is consistent (or satisfies certain properties), evaluate it
4. Keep on, as long as the blend evaluates as not 'good' enough

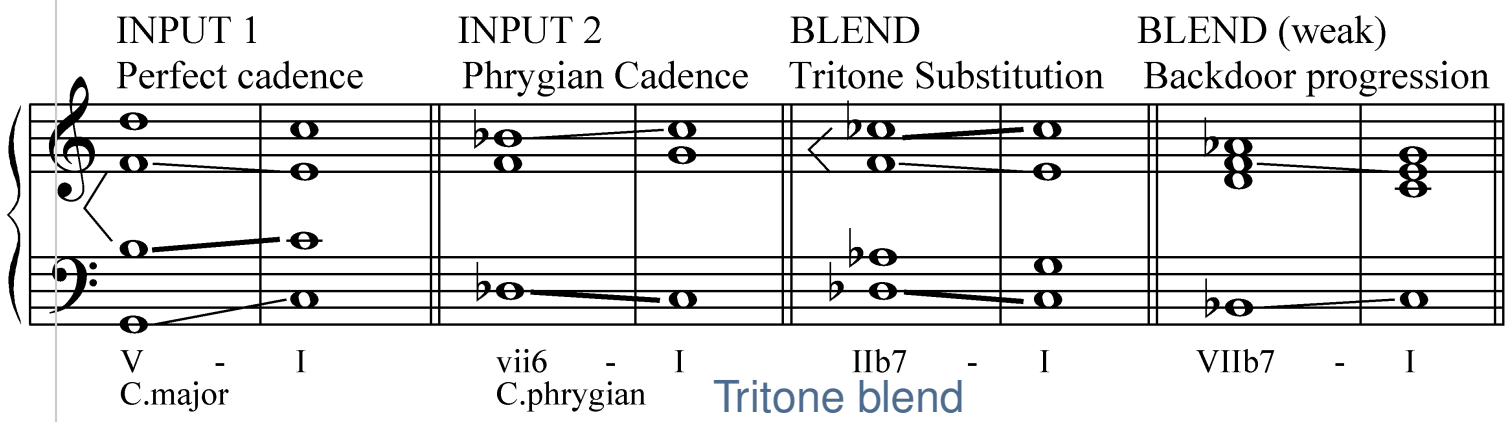


Use Case – Blending Cadences in Music



- **Inputs** are cadences used **centuries ago** – **blended** cadences are ones invented in **jazz** music centuries later!

Use Case – Blending Cadences in Music



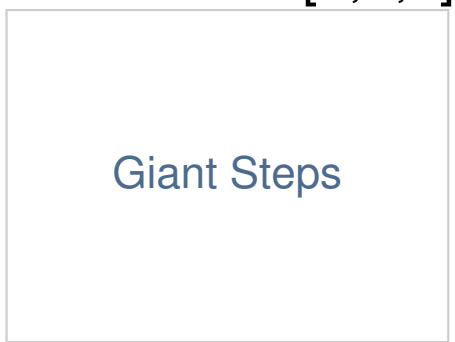
- **Inputs** are cadences used **centuries ago** – **blended** cadences are ones invented in **jazz** music centuries later!

Outlook – Cross-domain blending: Invent Coltrane changes

- Input Space 1:
  - Cyclic list of naturals
- Input Space 2:
  - A perfect cadence
- Generic Space:
  - root of perfect cadence = element of list
- Composition:
  - list of perfect cadences with roots at initial position
- Elaborated Blend:
  - Take only every n-th element in the cyclic list of 12
  - With $n = 4$: [0,4,8]

Outlook – Cross-domain blending: Invent Coltrane changes

- Input Space 1:
  - Cyclic list of naturals
- Input Space 2:
  - A perfect cadence
- Generic Space:
  - root of perfect cadence = element of list
- Composition:
  - list of perfect cadences with roots at initial position
- Elaborated Blend:
  - Take only every n-th element in the cyclic list of 12
  - With $n = 4$: [0,4,8]



  - 'Coltrane changes' – a milestone in Jazz
    | *G7* *C* | *B7* *E* | *Eb7* *Ab* |

References I

M. Boden. Creativity. In *Artificial Intelligence Handbook of Perception and Cognition*. Academic Press, 1996.

M. Eppe, R. Confalonieri, E. MacLean, M. Kaliakatsos, E. Cambouropoulos, M. Schorlemmer, and K.-U. Kühnberger. Computational invention of cadences and chord progressions by conceptual chord-blending. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015a.

M. Eppe, E. Maclean, O. Kutz, R. Confalonieri, M. Schorlemmer, and E. Plaza. ASP, Amalgamation, and the Conceptual Blending Workflow. In *Proceedings of the 13th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 2015b.

G. Fauconnier and M. Turner. Conceptual integration networks. *Cognitive Science*, 22(2):133–187, 4 1998.

G. Fauconnier and M. Turner. *The Way We Think: Conceptual Blending And The Mind's Hidden Complexities*. Basic Books, 2002.

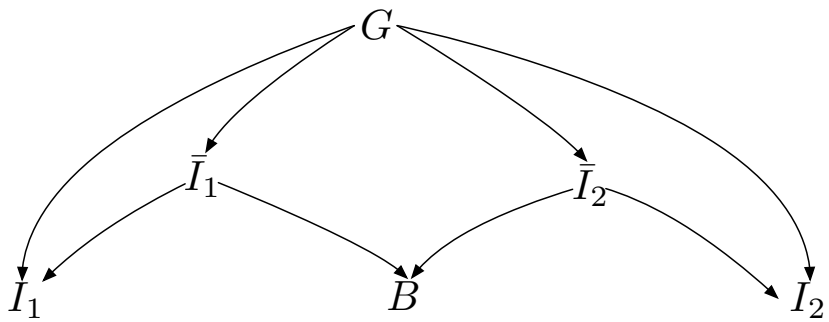
A. Koestler. *The Act of Creation*. Hutchinson and Co., 1964.

S. Ontañón and E. Plaza. Amalgams: A formal approach for combining multiple case solutions. In *ICCBR*, 2010.

M. Turner. *The Origin of Ideas*. Oxford University Press, 2014.

Amalgamation

- Our computational blending model is based on an *amalgamation* process
- Amalgamation originates from the notion of *amalgam* Ontañón and Plaza [2010] in case-based reasoning



- An *amalgam* of two input concepts is a new concept that combines parts from the original descriptions
  - Find *Generic Space* (G) of input concepts (commonalities) and try to combine non-common elements in I_1 and I_2
  - Often, input concepts I_1 and I_2 cannot be combined directly (inconsistency)
  - Input concepts have to be first *generalised* into I'_1 and I'_2
  - I'_1 and I'_2 can be finally blended to obtain a consistent B

Amalgamating perfect and Phrygian cadence $\mapsto$ tritone progression

spec PERFECTCADENCE =
CHORDPROGRESSION **then**

op	*c1Perf* : *Chord*	p:10
op	*c2Perf* : *Chord*	p:10
.	*succ*(*c1Perf*) = *c2Perf*	p:5
.	*keyNote*(*c1Perf*, 7)	p:2
.	*keyNote*(*c1Perf*, *11*)	p:3
.	*keyNote*(*c1Perf*, 2)	p:1
.	*keyNote*(*c1Perf*, 5)	p:2
. . .		

end

spec PHRYGIACADENCE =
CHORDPROGRESSION **then**

op	*c1Phry* : *Chord*	p:10
op	*c2Phry* : *Chord*	p:10
.	*succ*(*c1Phry*) = *c2Phry*	p:5
.	*keyNote*(*c1Phry*, 10)	p:1
.	*keyNote*(*c1Phry*, *1*)	p:3
.	*keyNote*(*c1Phry*, 5)	p:2
. . .		

end

Blending workflow:

1. Generalization by merging: c1Perf and c1Phry $\mapsto$ c2Perf and c2Phry
2. Generalization by removing axioms: remove notes that cause dissonances
3. Composition: unify remaining notes
4. Completion and elaboration: determine root and deduce additional notes
5. Evaluation: Check consistency and apply optimality principles

Blend Evaluation – Optimality Principles

Blend evaluation in terms of computational characterisation of the optimality principles proposed by Fauconnier and Turner [2002]

- Unpacking, web, topology and integration:
  - Consistency
  - Keep as much information as possible from the input spaces
- Vital relations: blends should maximize common relations, as a means to compress the structure of the input spaces
- Double-scope property: the amount of information from the input specifications should be balanced

Evaluation Principles – Implementation

1. Amount of information

$$infoValue(\mathfrak{s}) = \sum_{e \in \mathfrak{s}} prio(e) \tag{1}$$

   Sum of priority values of the individual elements

2. Compression

$$compression(\mathfrak{c}) \tag{2}$$

   Sum of priorities of merged elements from generalized input specs

3. Double-scopeness

$$imbalance(\mathfrak{c}) = \frac{abs(infoValue(\mathfrak{s}_1) - infoValue(\mathfrak{s}_2))}{2} \tag{3}$$

   Half the difference of amount of information from the generalized input spaces.

($\mathfrak{c}$: potential blend ; $\mathfrak{s}$: generalized input specification)

Generalizing and blending NAT and LIST

Generalizing NAT:

$exec(rename(Nat, L, \text{LIST}), \text{NAT}, 0), exec(rename(zero, nil, \text{LIST}), \text{NAT}, 1), \ldots,$
$exec(rename(sum, rev, \text{LIST}), \text{NAT}, 4), exec(rename(qsum, qrev, \text{LIST}), \text{NAT}, 5), \ldots,$
$exec(rm(4), \text{NAT}, 7), exec(rm(5), \text{NAT}, 8), exec(rm(1), \text{NAT}, 9),$
$exec(rm(2), \text{NAT}, 10), exec(rm(c), \text{NAT}, 11), exec(rm(NL), \text{NAT}, 12)\}$

Generalizing LIST:

$exec(rm(10), \text{LIST}, 0), exec(rm(11), \text{LIST}, 1), exec(rm(9), \text{LIST}, 2), exec(rm(7), \text{LIST}, 3)$

Colimit of NAT @ $t = 11$ with LIST @ $t = 0$ gives the original list with the Eureka lemma $app(rev(x), y) = qrev(x, y)$