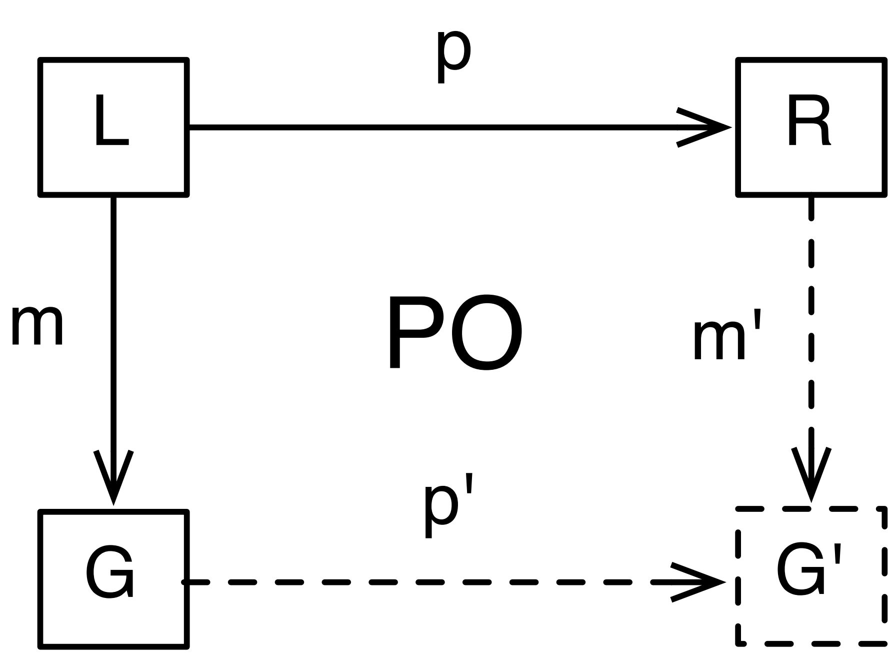
UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

BX with Triple Graph Grammars

PART 3: FORMAL PROPERTIES

Formal Notation



$G \xRightarrow{p@m} G'$

compact representation of a rule application (aka **direct derivation**)

$G \overset{p}{\Rightarrow} G'$

even more compact

$G \xRightarrow{p_1} G_1 \xRightarrow{p_2} \cdots \xRightarrow{p_n} G_n$

derivation of length n > 1

$G \overset{*}{\Rightarrow} G_n$

compact representation of derivation

Formal Notation

$$G \overset{p_1}{\Longrightarrow} G_1 \overset{p_2}{\Longrightarrow} \cdots \overset{p_n}{\Longrightarrow} G_n$$

derivation of length n > 1

$$G \overset{*}{\Rightarrow} G_n$$

compact representation of derivation

compact notation for derivation with rules of a TGG

$$G \Rightarrow^* G' \in TGG \Leftrightarrow$$
$$\exists\, G \overset{p_1}{\Longrightarrow} G_1 \overset{p_2}{\Longrightarrow} \cdots \overset{p_n}{\Longrightarrow} G_n, \forall i \in \{1, \ldots, n\}, p_i \in TGG$$

(Transformation) Correctness: Intuition

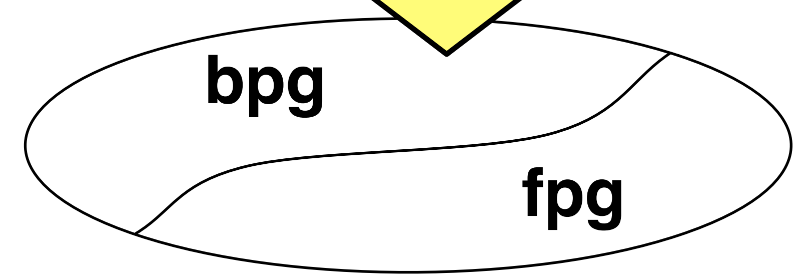
$$\mathbf{A} \xleftrightarrow{\mathbf{R}} \mathbf{B}$$

\+

TGG

use rules specified by the user to characterise a subset of “consistent” triples

intuition: (fpg, bpg) should be consistent with its underlying TGG

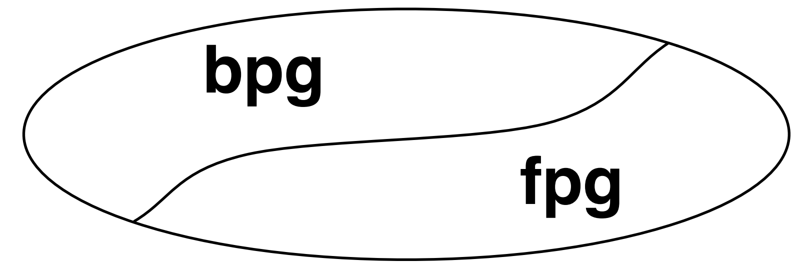


(Transformation) Correctness: Notation

$$\mathbf{A} \xleftrightarrow{\mathbf{R}} \mathbf{B}$$

+

TGG



$$C = \{A \xleftrightarrow{r} B \mid \exists\, \emptyset \overset{*}{\Rightarrow} A \xleftrightarrow{r} B \in TGG\}$$
$$\subseteq \mathbf{R}$$

use rules specified by the user to characterise a subset of “consistent” triples

just compact notation for consistent triples

$$A \xleftrightarrow{r} B \in C \Leftrightarrow A \xleftrightarrow{r:C} B$$

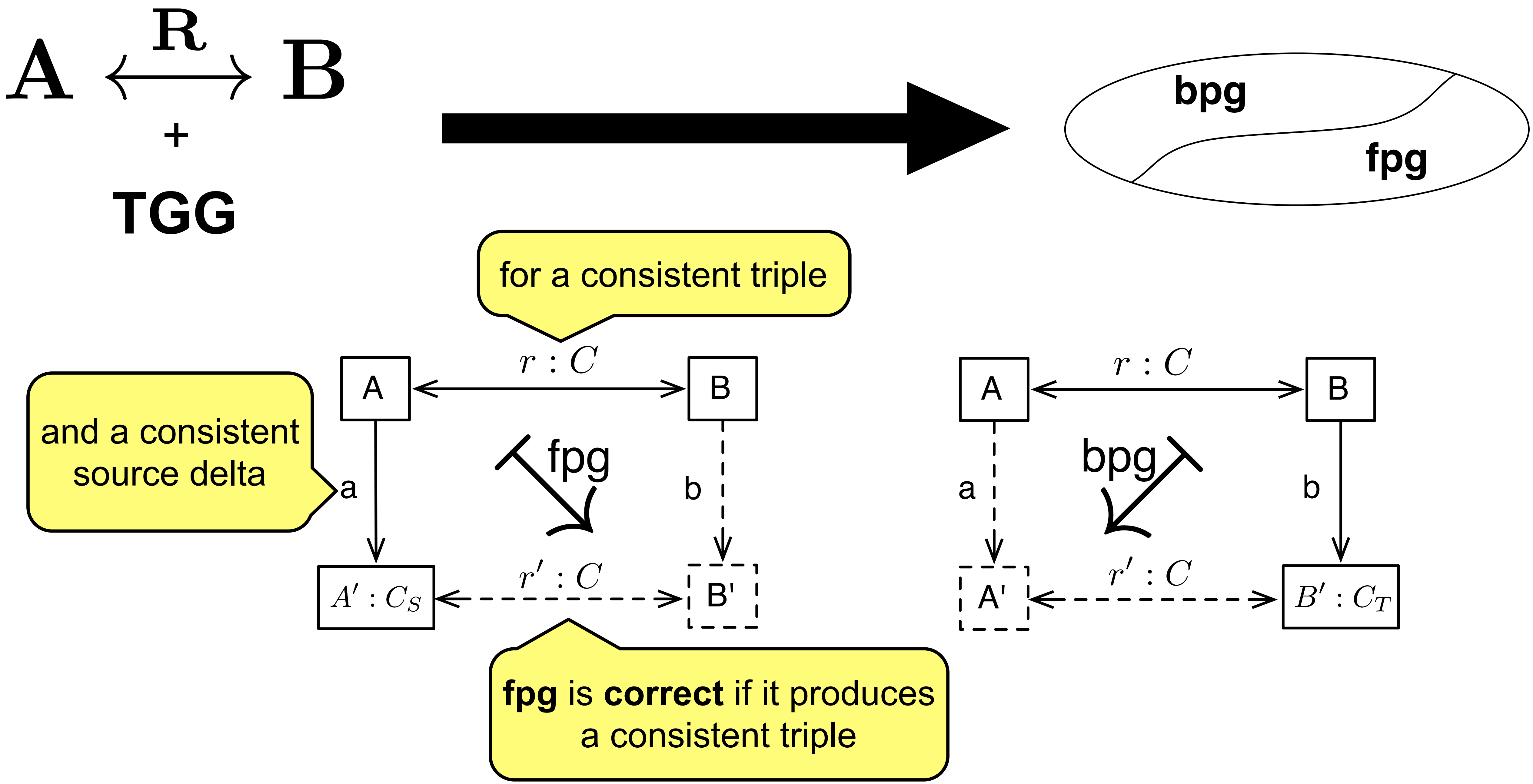
$$C_S = \{A \mid \exists\, A \xleftrightarrow{r:C} B\}$$

this is the set of all consistent source models (consistent target models analogously)

$$A : C_S \xrightarrow{a} A' : C_S$$

let’s call such a source delta **consistent**

(Transformation) Correctness: Laws



(Transformation) Completeness

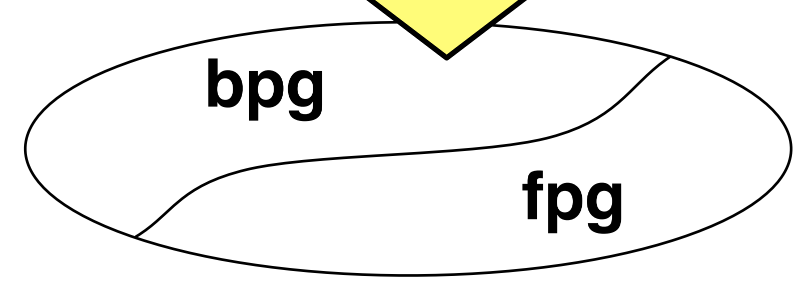
$$\mathbf{A} \xleftrightarrow{\mathbf{R}} \mathbf{B}$$

+

TGG

remember: the supplied TGG only covers a small subset of all deltas!

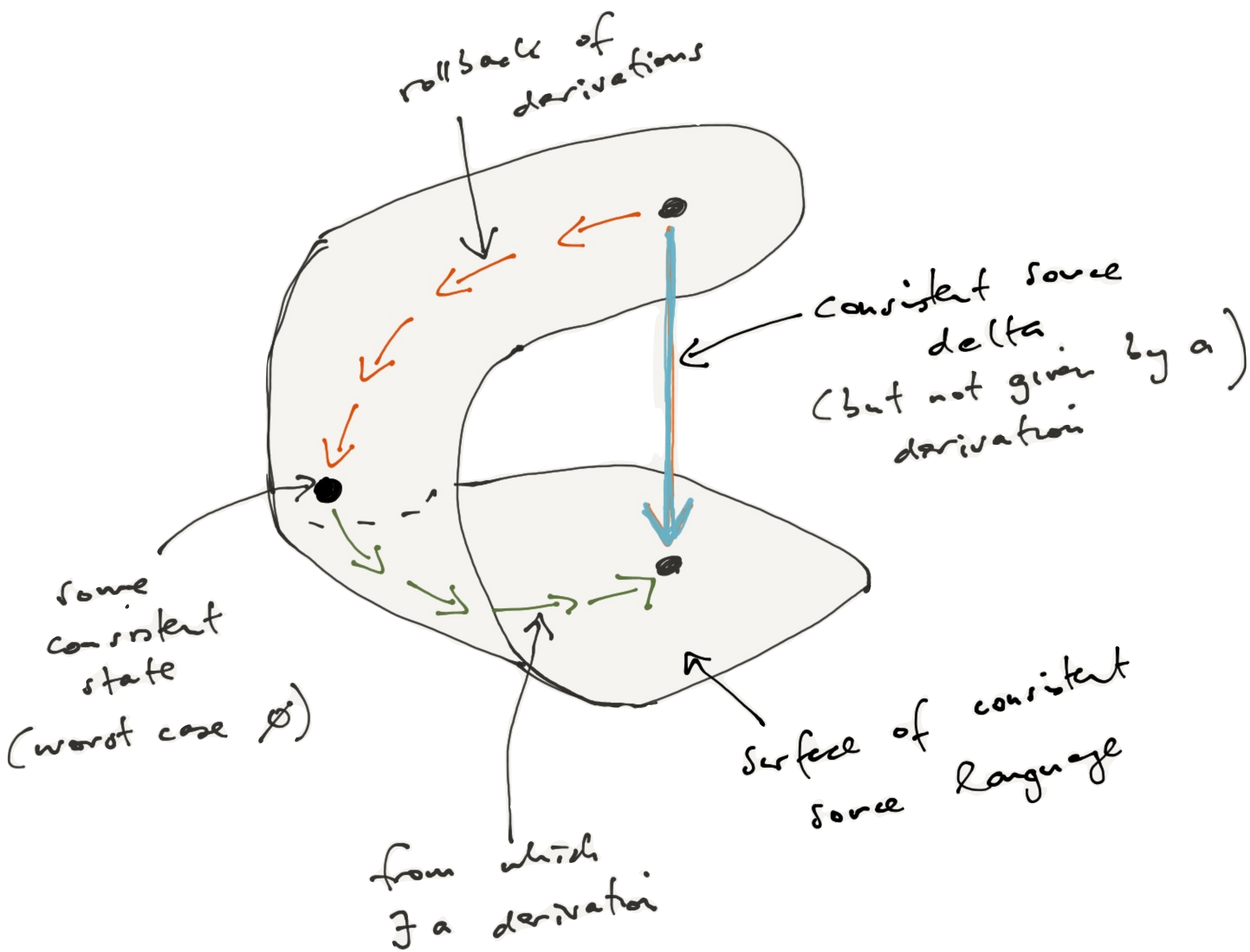
intuition: (fpg, bpg) should be defined for **any** consistent delta



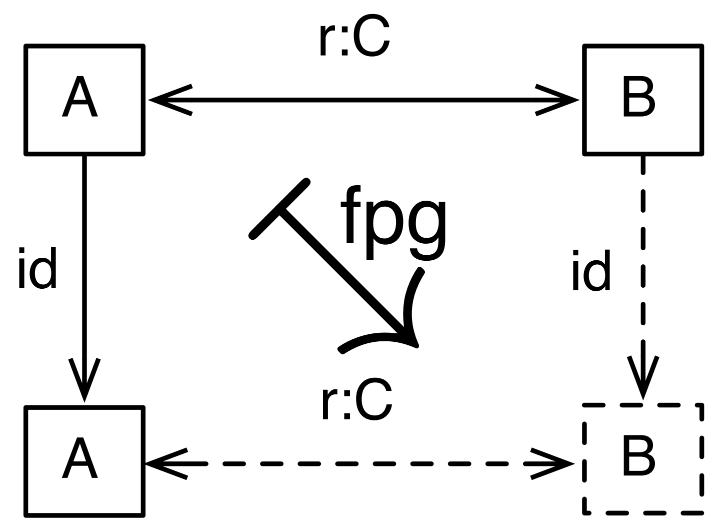
formally, completeness just means that (fpg, bpg) must be **total** on all consistent source/target models and consistent source/target deltas

Consistent source delta $\hat{=}$ Arrow connecting points on the surface

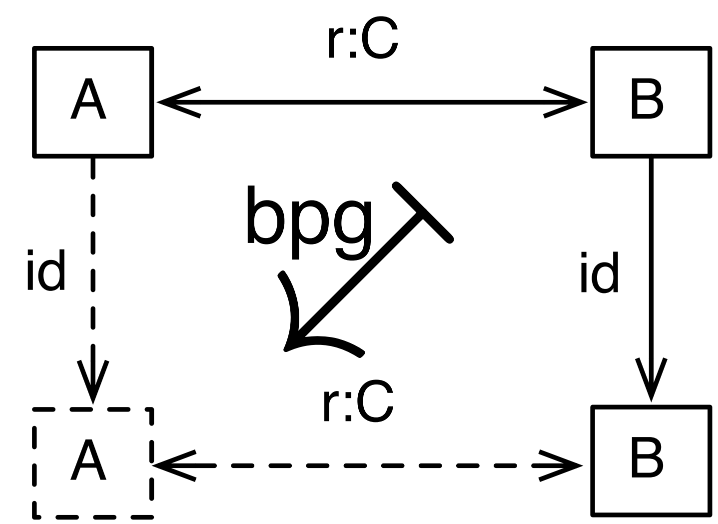
Completeness $\hat{=}$ Synchroniser always finds a "path" along the surface for any consistent source delta



Stability

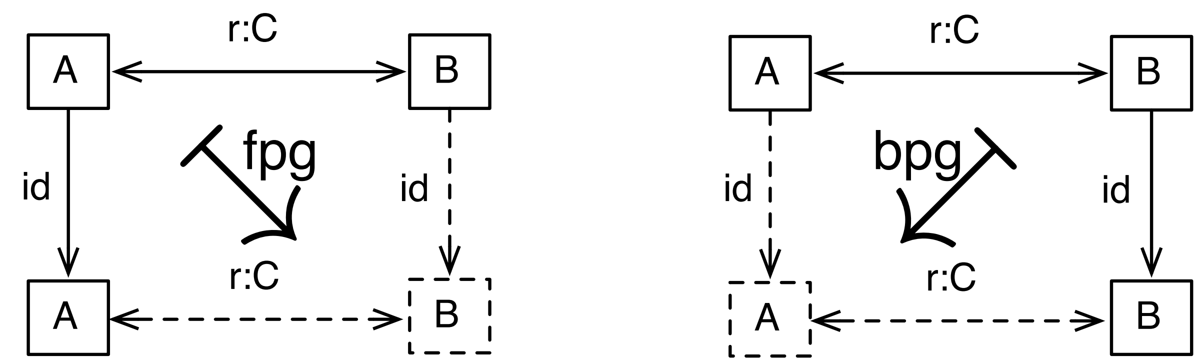


don’t do anything for the “idle” delta

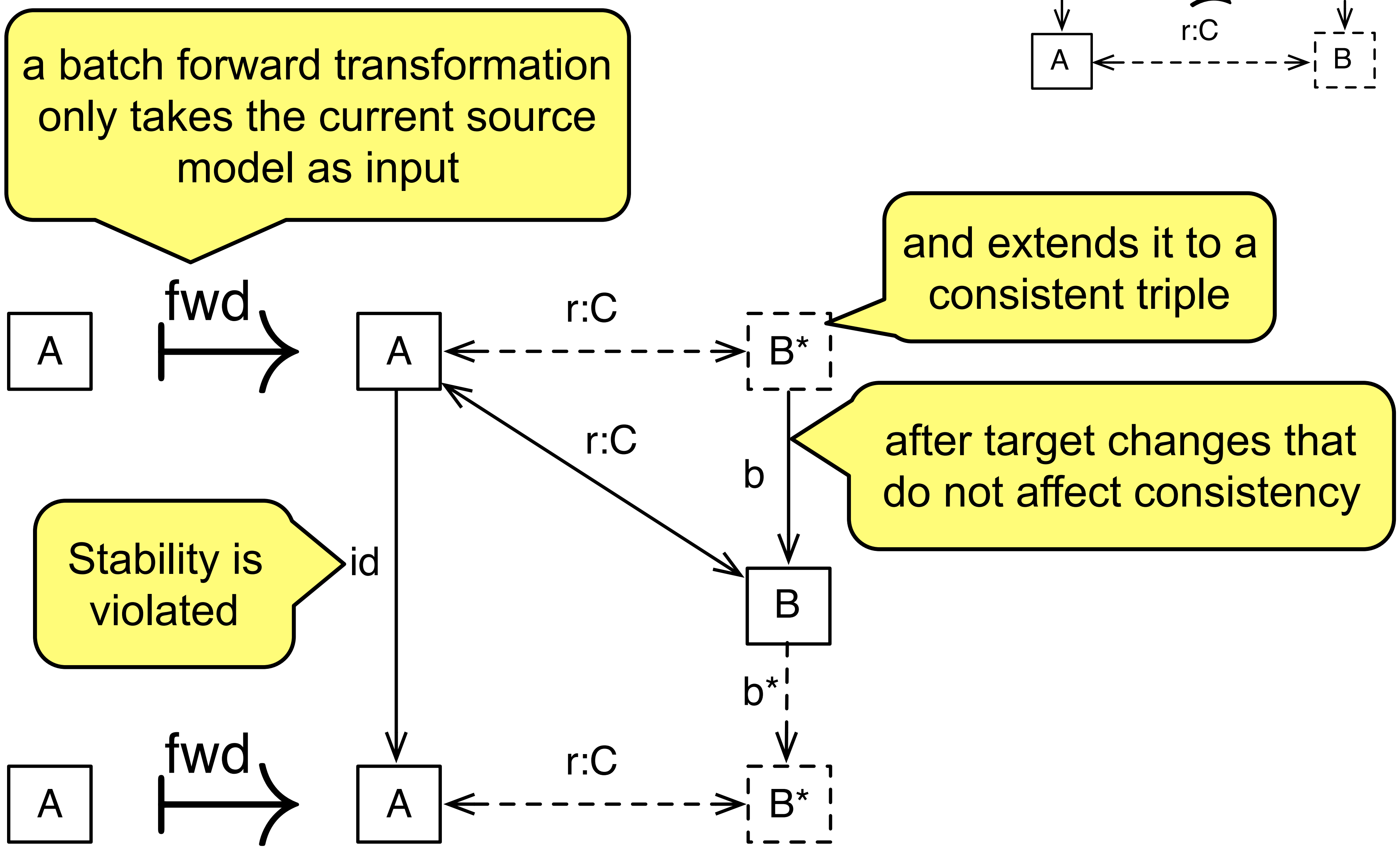


sounds trivial, but it rules out “batch mode” TGG tools

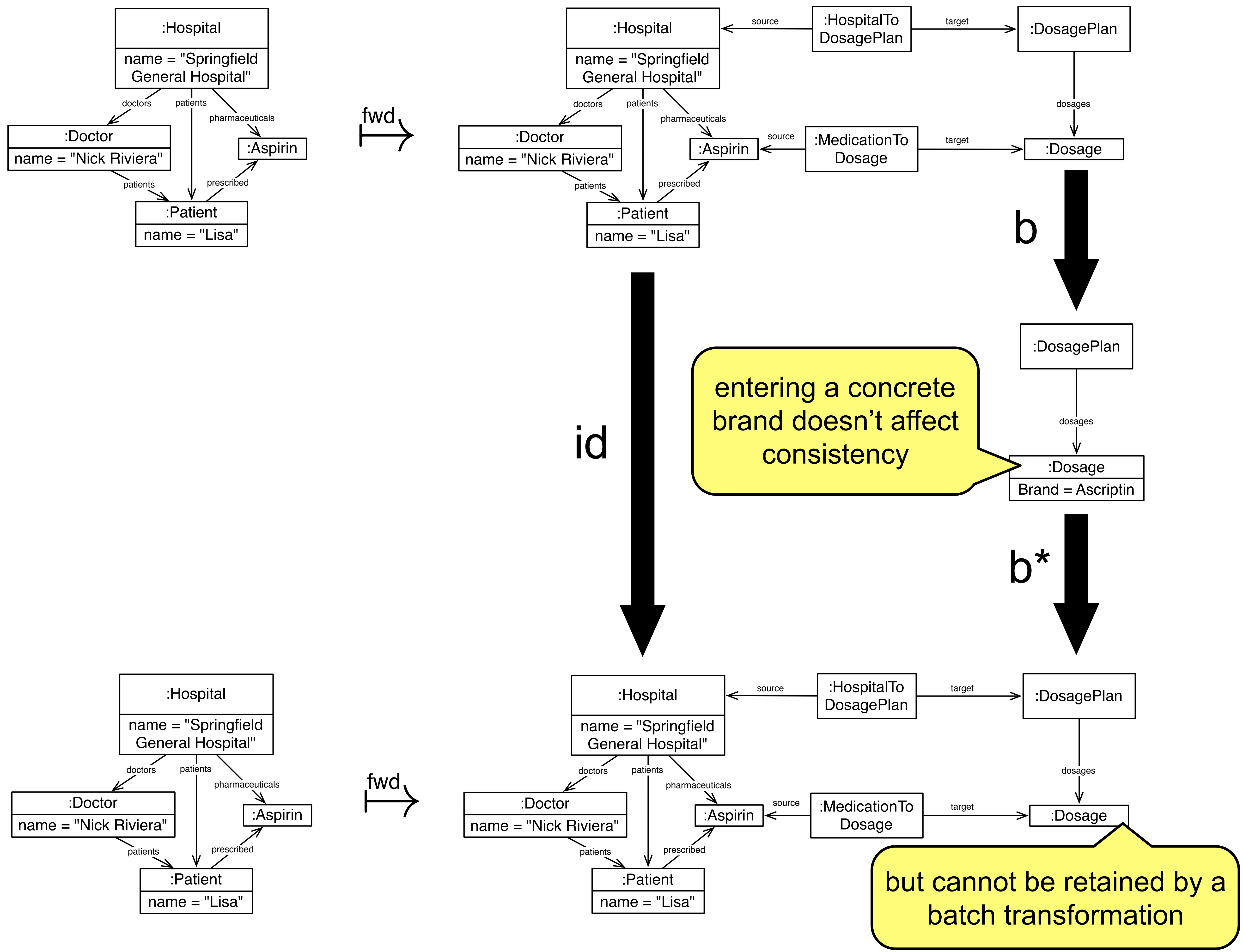
Stability



r:C
A
B
fpg
id
id
r:C
A
B
r:C
A
B
bpg
id
id
r:C
A
B



a batch forward transformation only takes the current source model as input
and extends it to a consistent triple
after target changes that do not affect consistency
Stability is violated
A
fwd
A
r:C
B*
r:C
b
id
B
b*
A
fwd
A
r:C
B*



:Hospital
name = "Springfield General Hospital"
doctors
patients
pharmaceuticals
:Doctor
name = "Nick Riviera"
:Aspirin
patients
prescribed
:Patient
name = "Lisa"
fwd
source
:HospitalTo DosagePlan
target
:DosagePlan
dosages
:MedicationTo Dosage
:Dosage
b
id
entering a concrete brand doesn't affect consistency
Brand = Ascriptin
b*
but cannot be retained by a batch transformation

Other laws

1. Hippocraticness
2. (Weak) Undoability
3. (Weak) Invertibility
4. Functional Behaviour
5. Domain Correctness
6. Domain Completeness
7. Local Completeness
8. …

Other laws

1. Hippocraticness
2. (Weak) Undoability
3. (Weak) Invertibility
4. Functional Behaviour
5. Domain Correctness
6. Domain Completeness
7. Local Completeness
8. …

Diskin et al. calls such SDLs “well-behaved”

Other laws

1. Hippocraticness
2. (Weak) Undoability
3. (Weak) Invertibility
4. Functional Behaviour
5. Domain Correctness
6. Domain Completeness
7. Local Completeness
8. …

in **general** TGG-based synchronisation does not obey any of these laws …

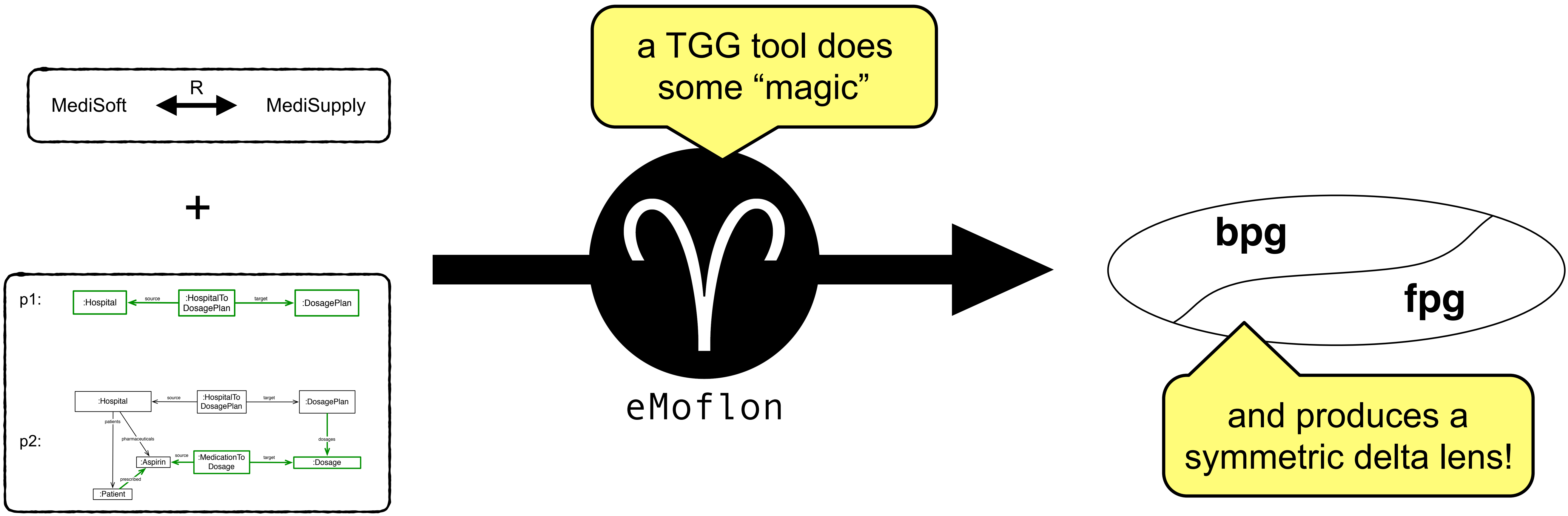
… but suitable **restrictions** can be posed to determine adequate subclasses of TGGs

TGGs offer a “playground” for exploring formal properties and how to guarantee them (statically or dynamically)

BX with Triple Graph Grammars

PART 4: FROM TGGS TO SDLS

What do TGG tools do?

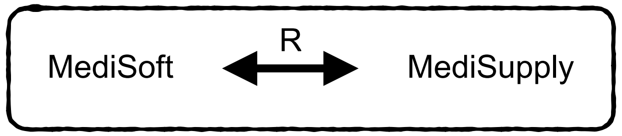


Notation

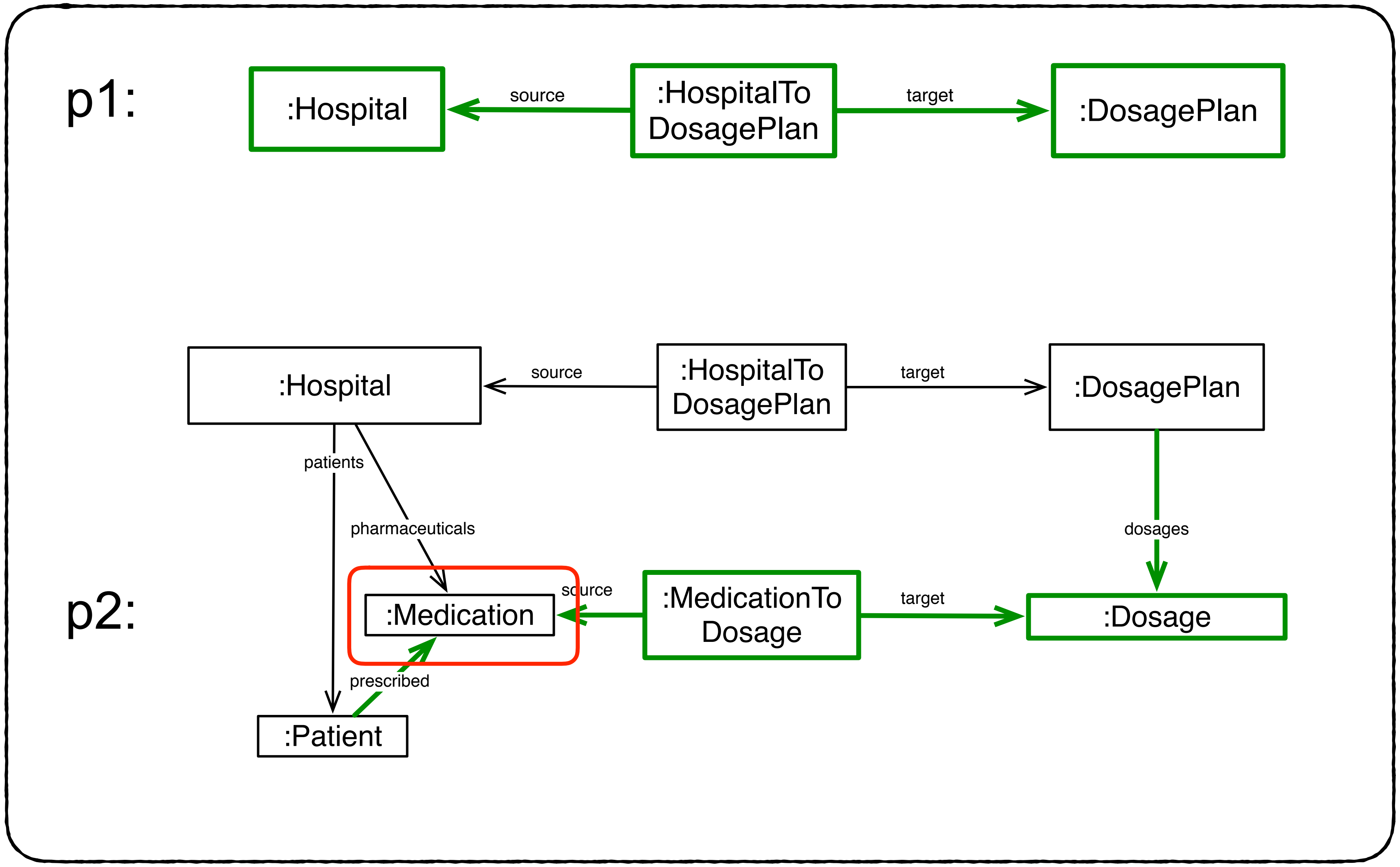
1. All arrows are **monomorphic** (injective) (my head hurts otherwise)

2. Colours indicate deletion (red) and creation (green)

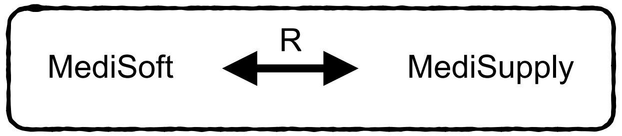
Running Example



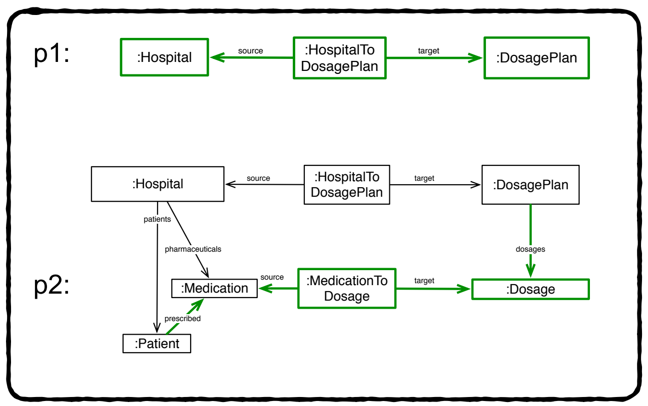
+

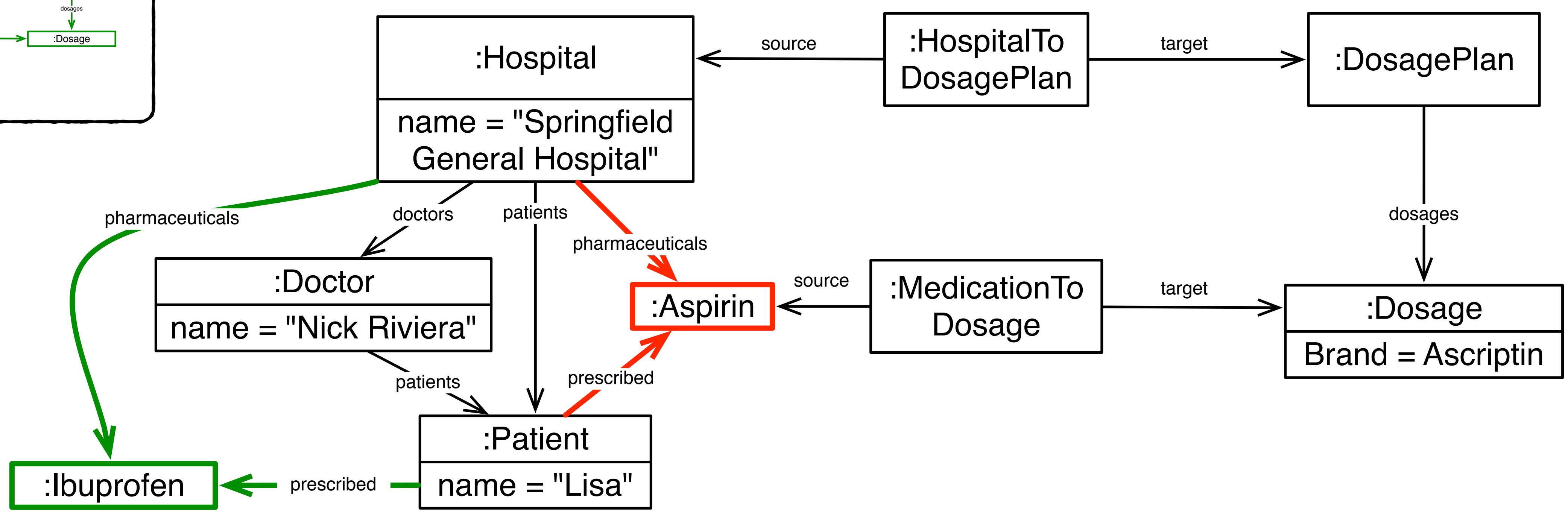


Running Example



+

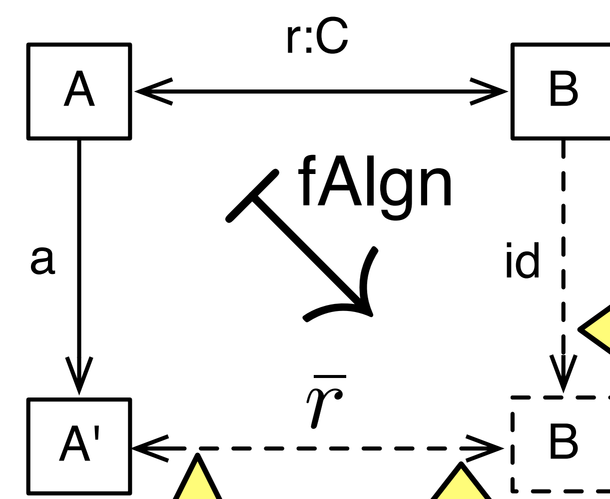




What do TGG tools do?

basic idea:
realise correct fpg
in three steps

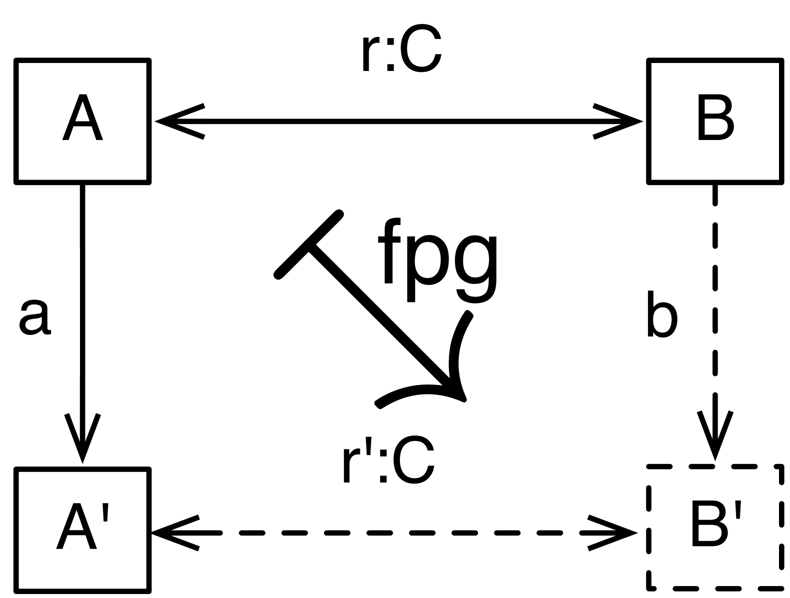
1. (Re-)Alignment:



this does not change the target model

note that the result is not necessarily consistent!

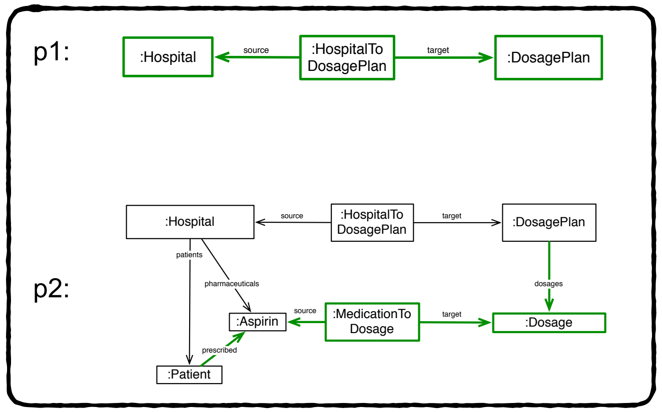
fAlgn recomputes correspondence links



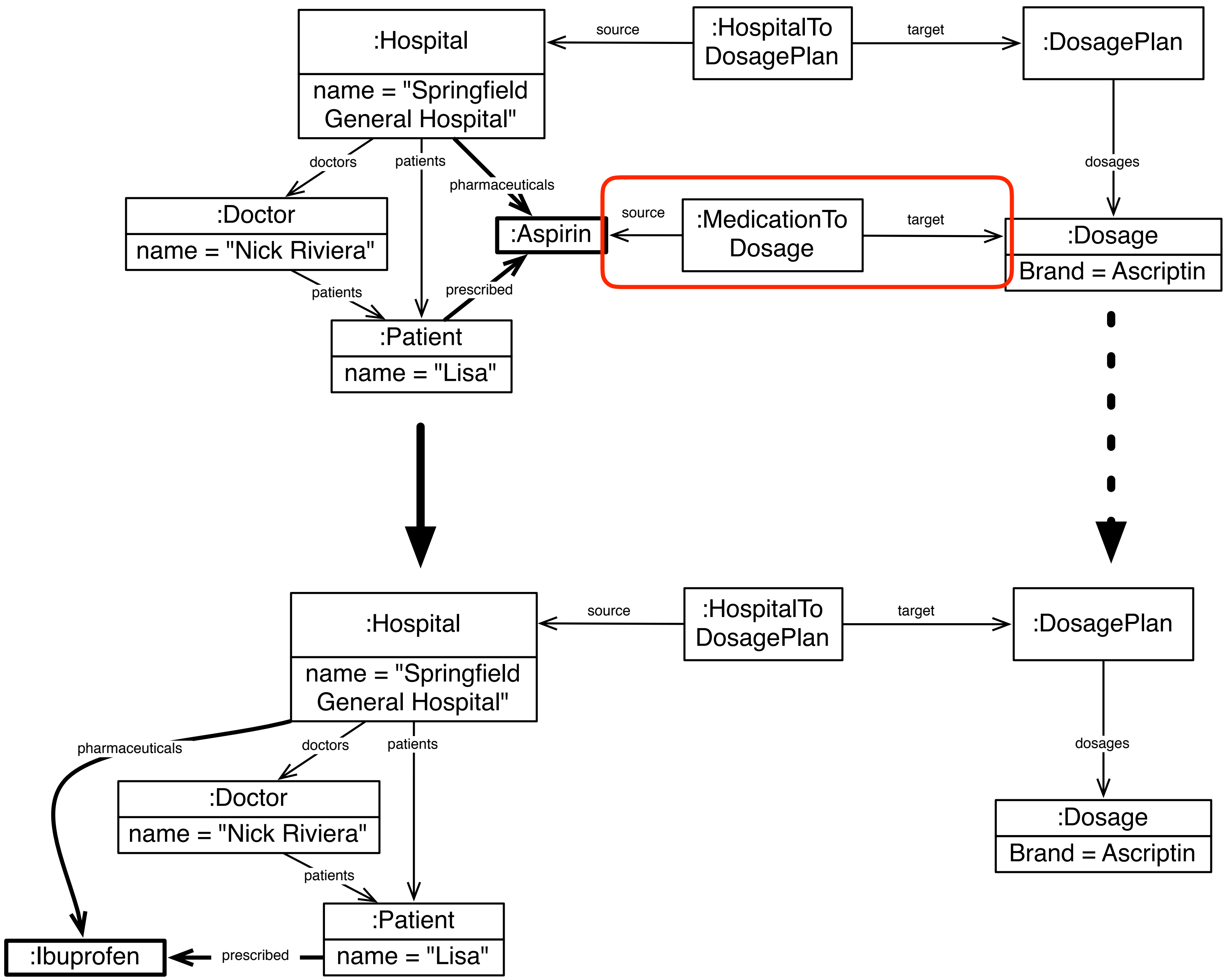
Running Example: Re-Alignment

MediSoft $\xleftrightarrow{R}$ MediSupply

\+

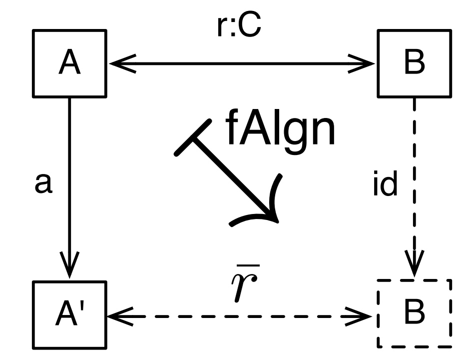


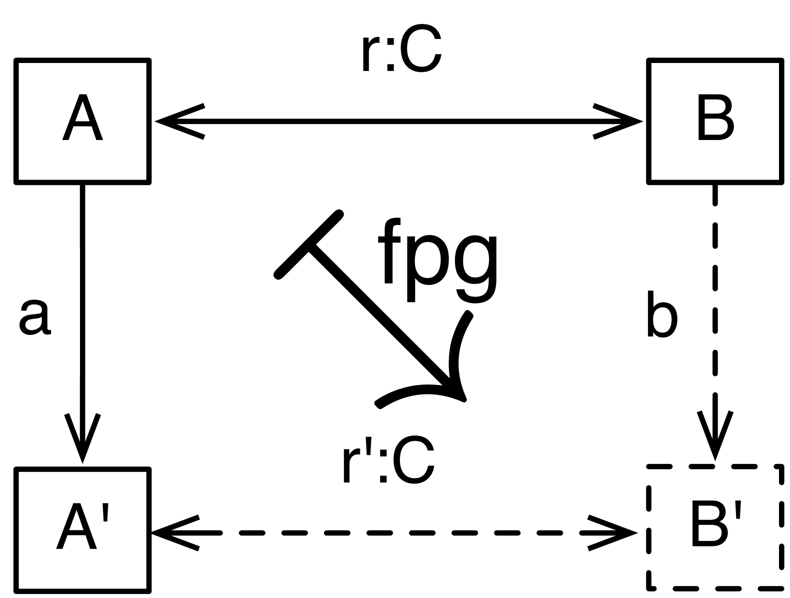
$$
\begin{tikzcd}
A \arrow[rr, leftrightarrow, "r:C"] \arrow[d, "a"'] \arrow[drr, phantom, "\text{fAlgn}"] & & B \arrow[d, dashed, "id"] \\
A' \arrow[rr, leftrightarrow, dashed, "\bar{r}"'] & & B
\end{tikzcd}
$$



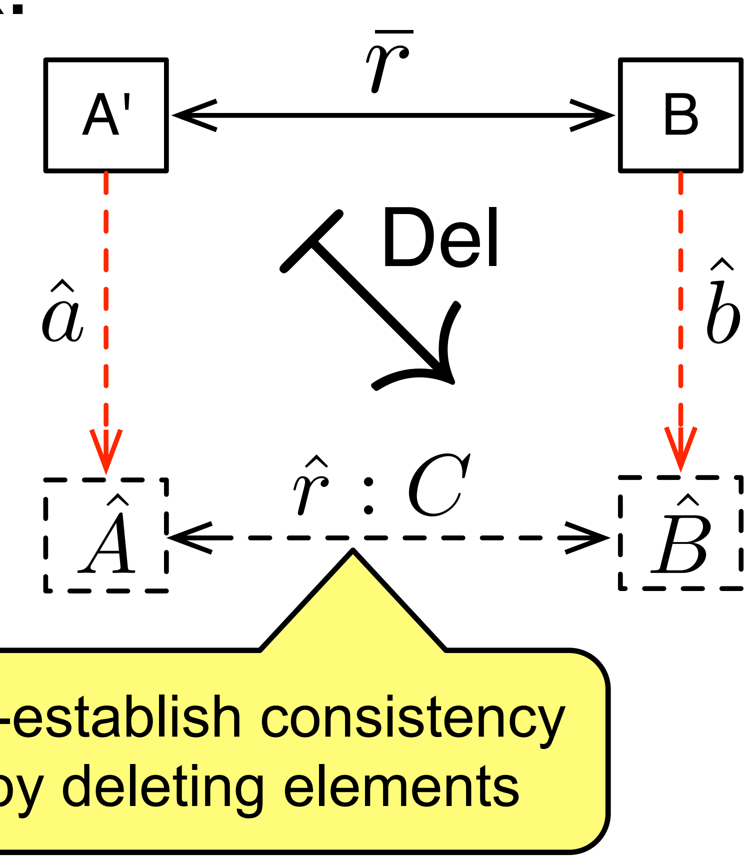
What do TGG tools do?

1. (Re-)Alignment:

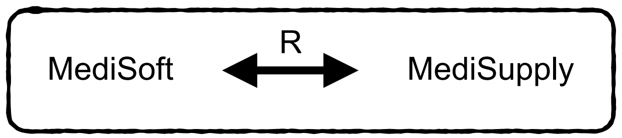




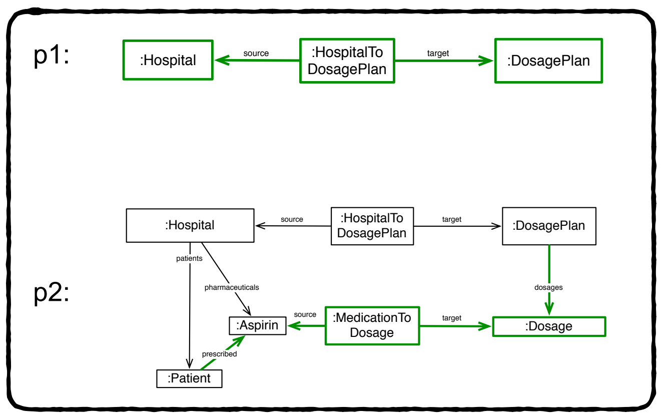
2. Rollback:

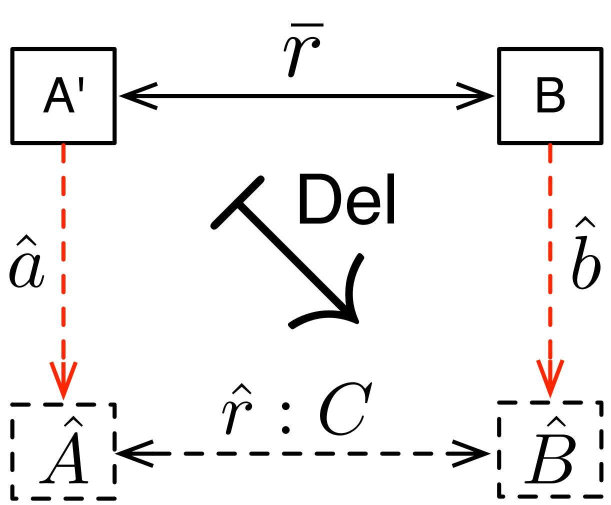


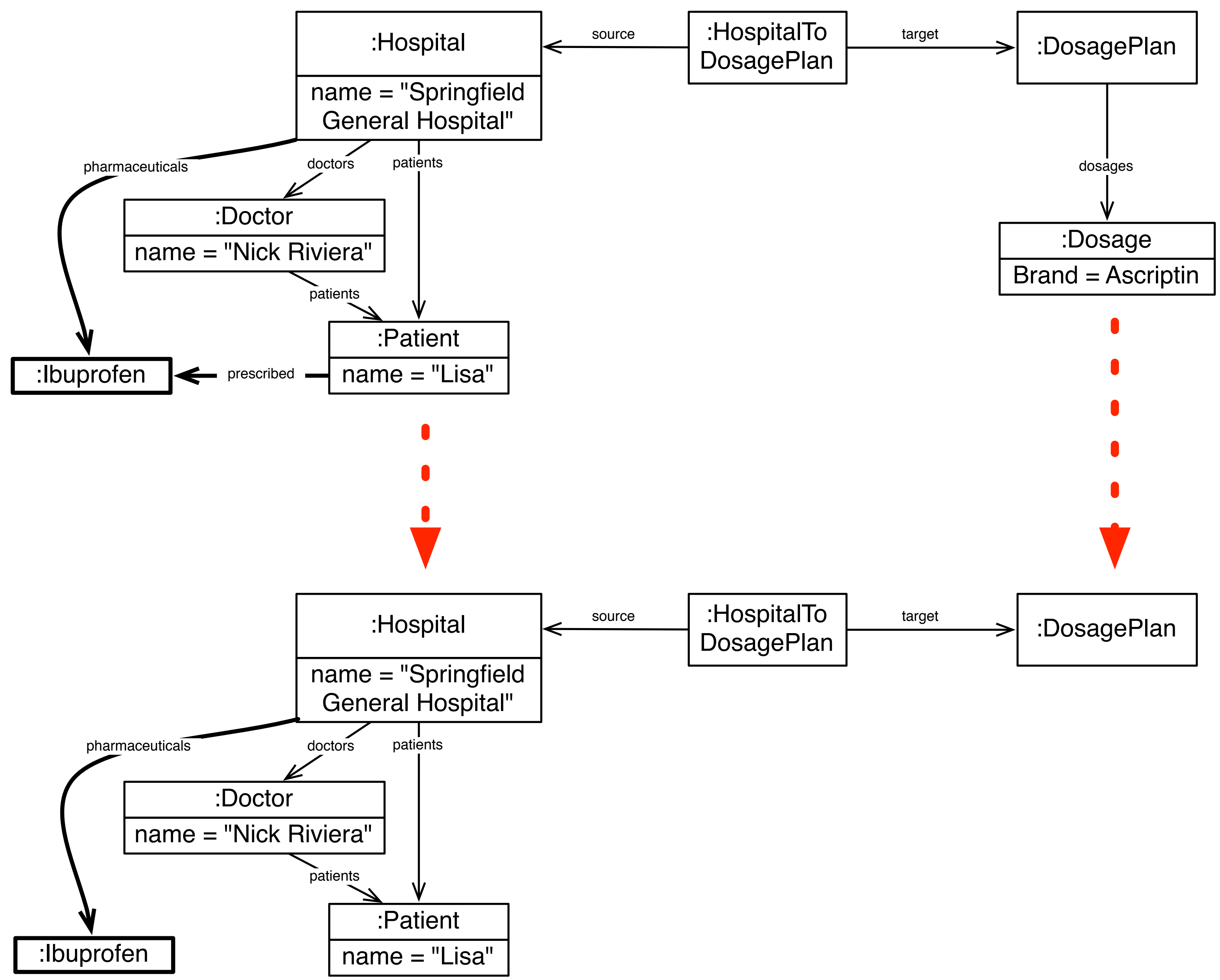
Running Example: Rollback



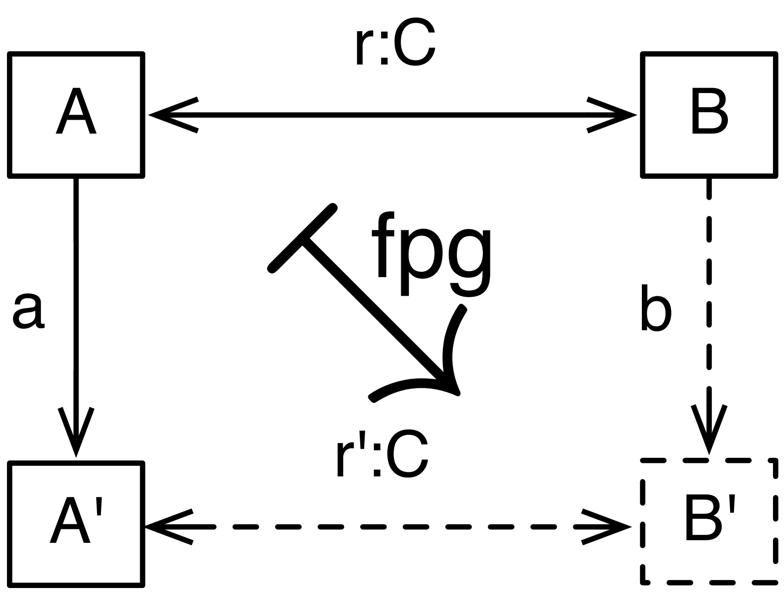
+



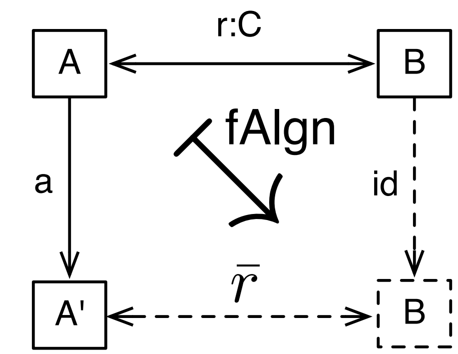




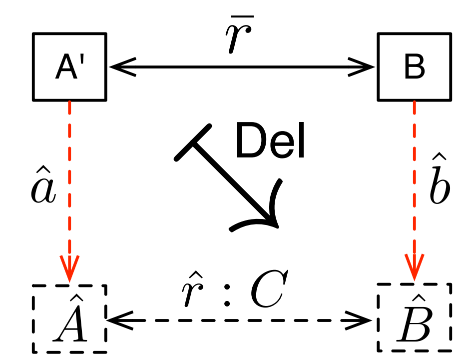
What do TGG tools do?



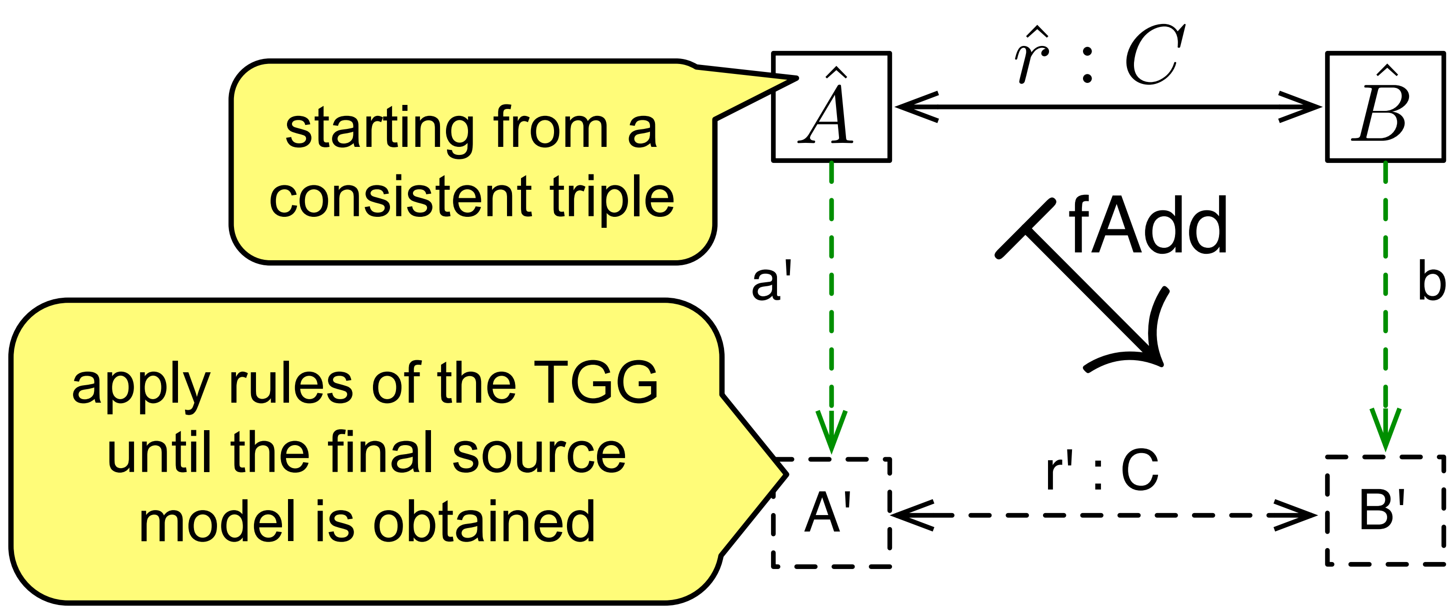
1. (Re-)Alignment:



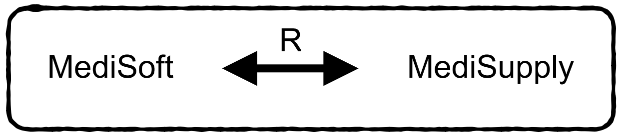
2. Rollback:



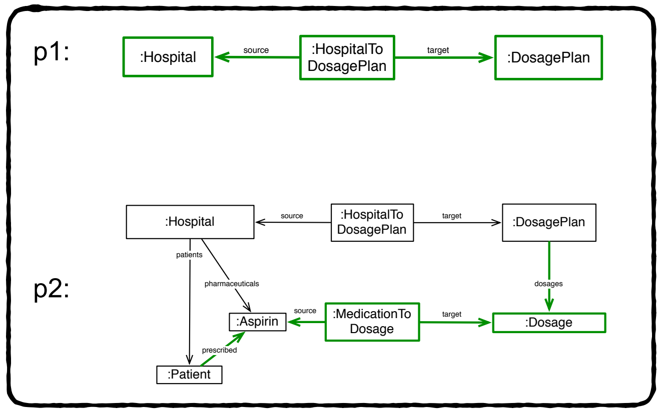
3. (Re-)Translation:

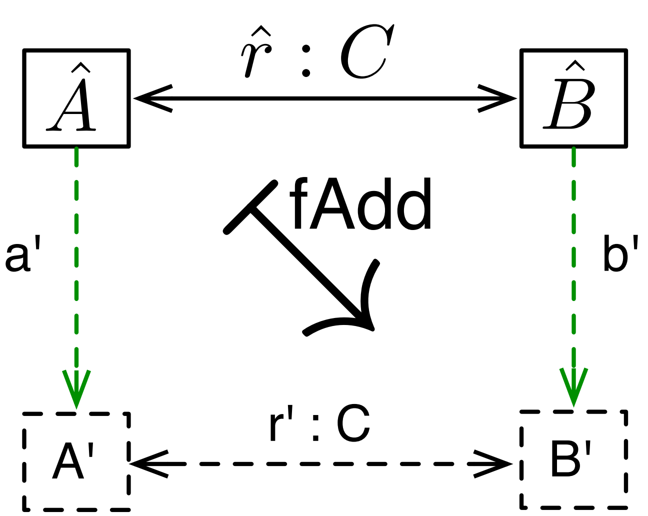


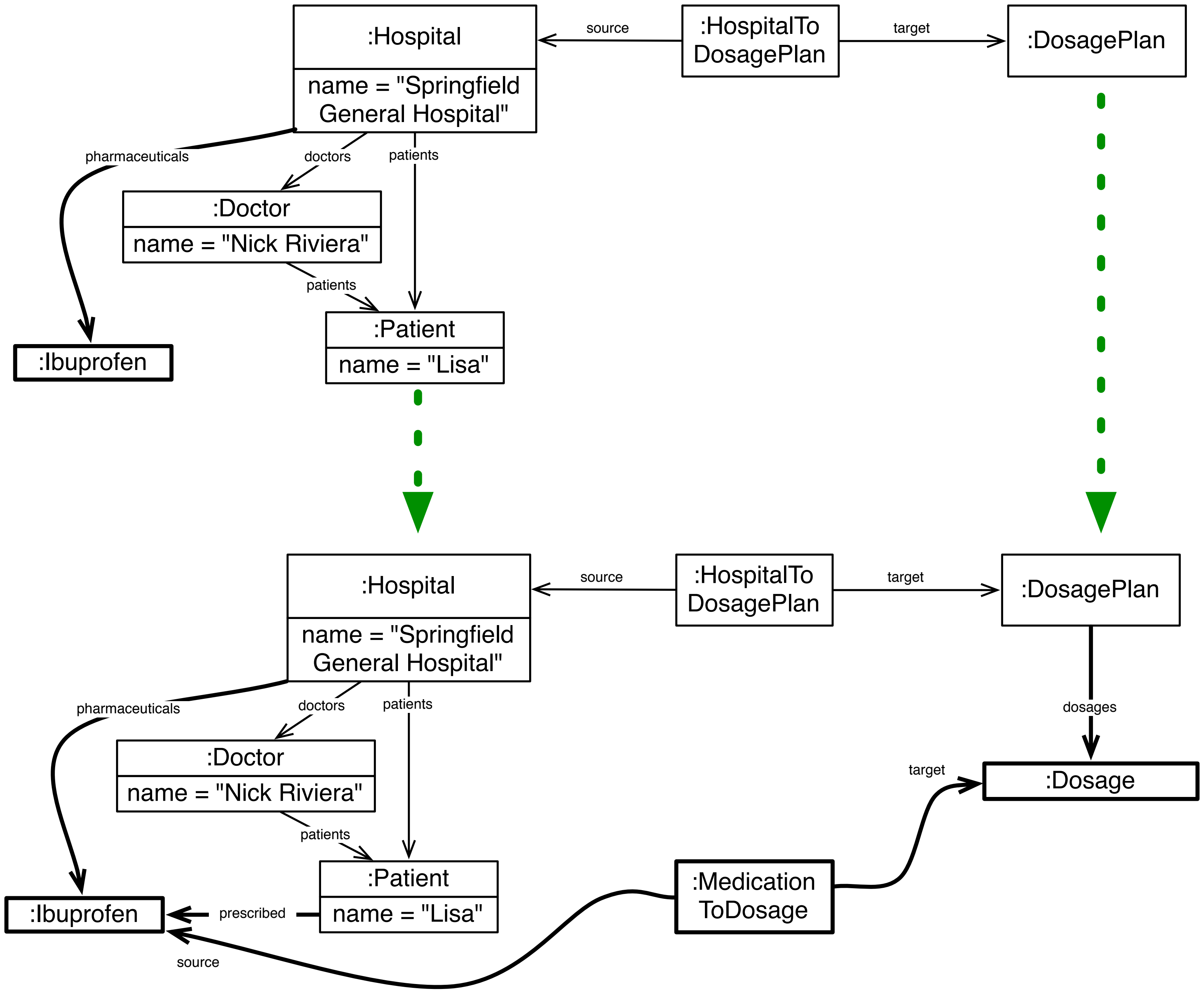
Running Example: Re-Translation



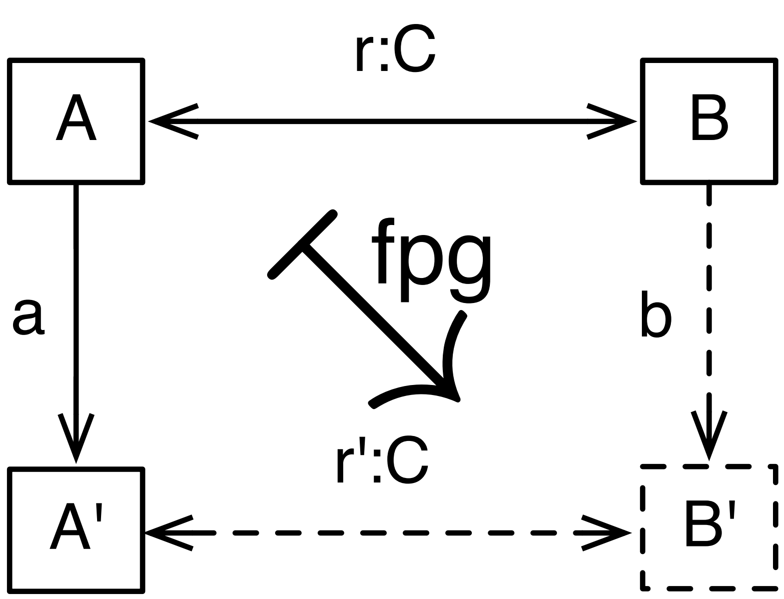
+



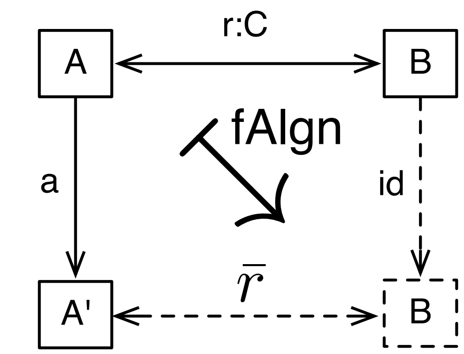




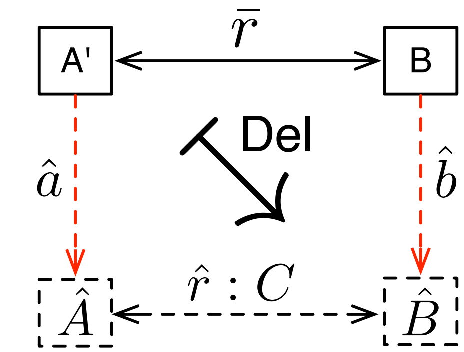
What do TGG tools do?



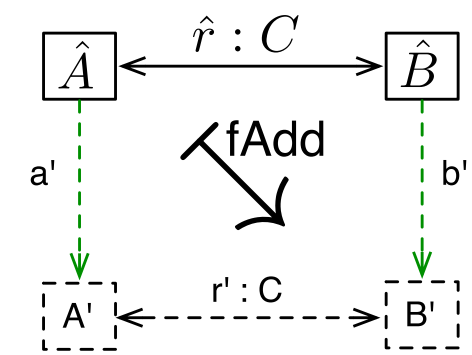
1. (Re-)Alignment:



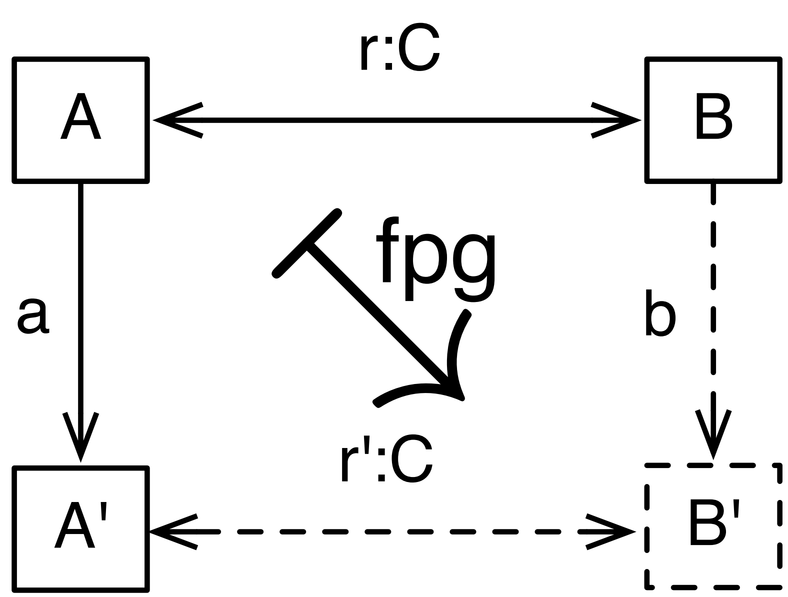
2. Rollback:

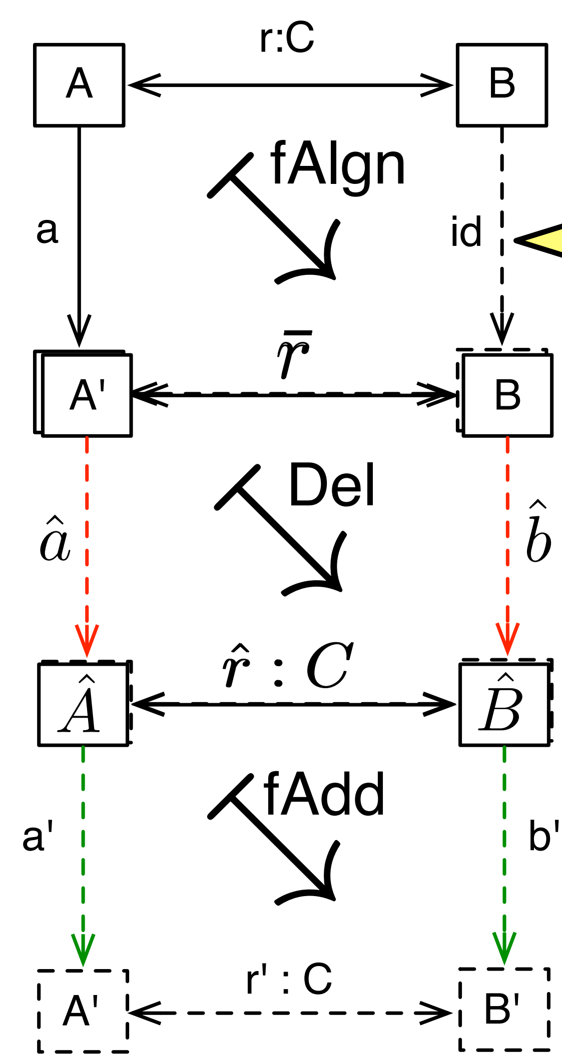


3. (Re-)Translation:

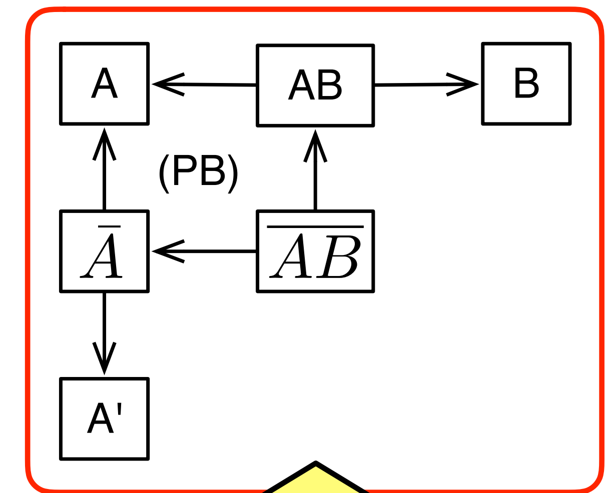


Some remarks on implementation



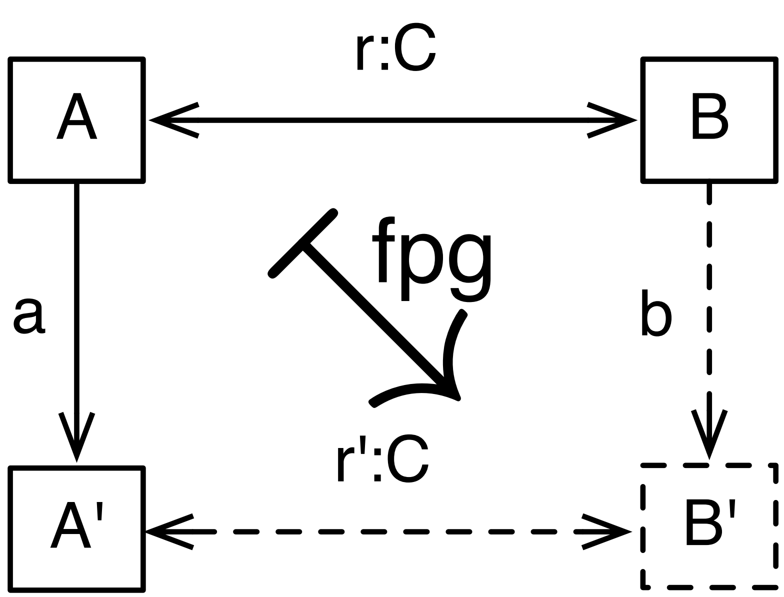


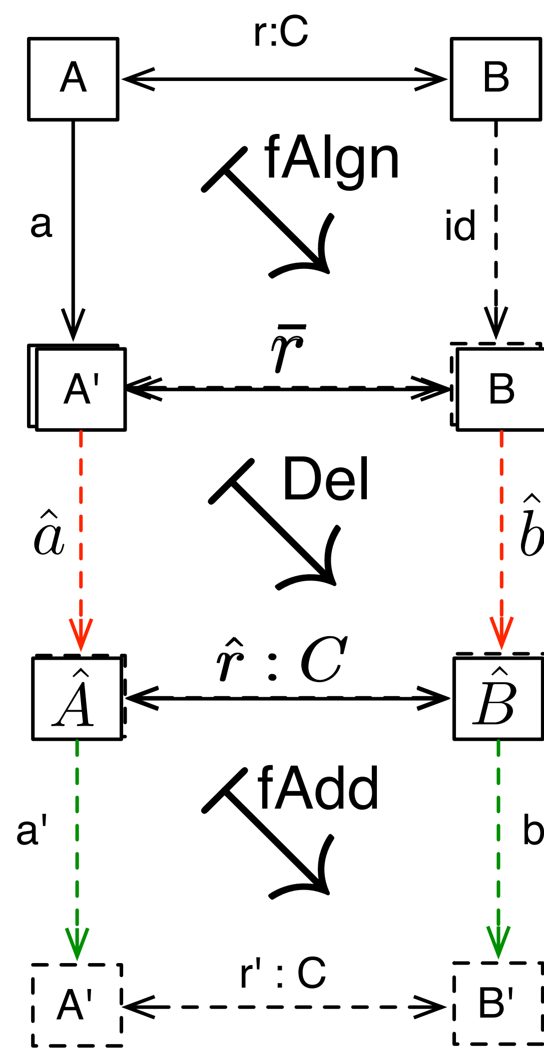
easy: TGG tools represent correspondence links explicitly so can just delete “dangling” links (formally a pullback)



arrows in this diagram are monic typed graph morphisms (and not deltas!)

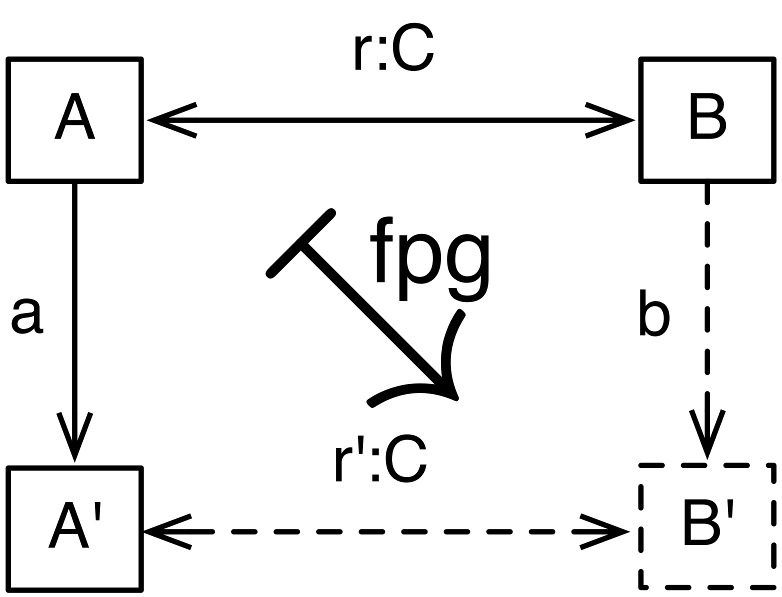
Some remarks on implementation

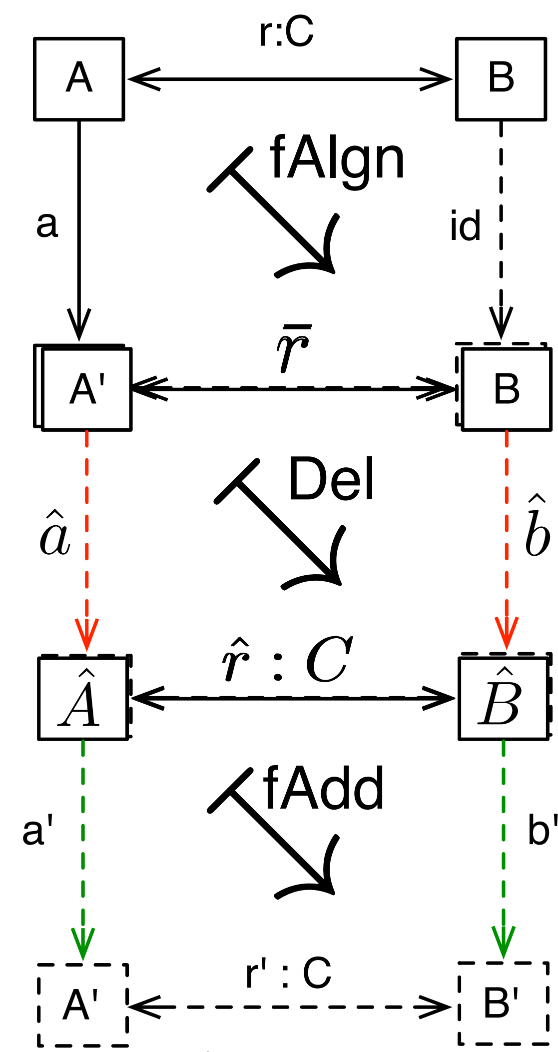




hard: requires a complete remarking of all elements (very inefficient), most TGG tools employ some kind of optimisation technique

Some remarks on implementation



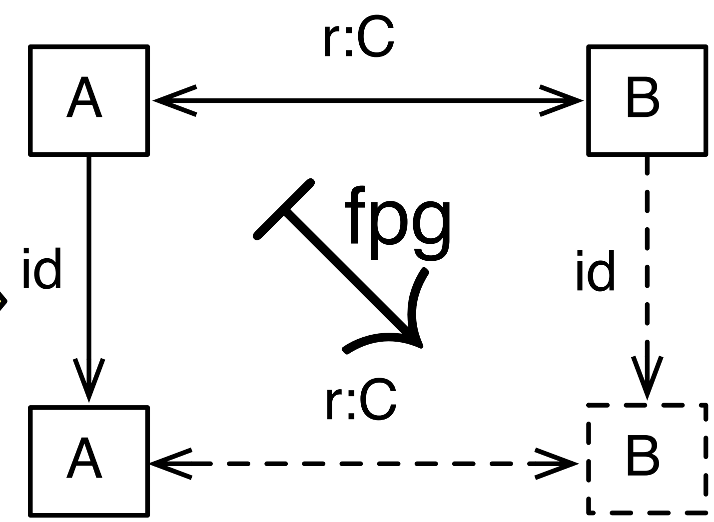


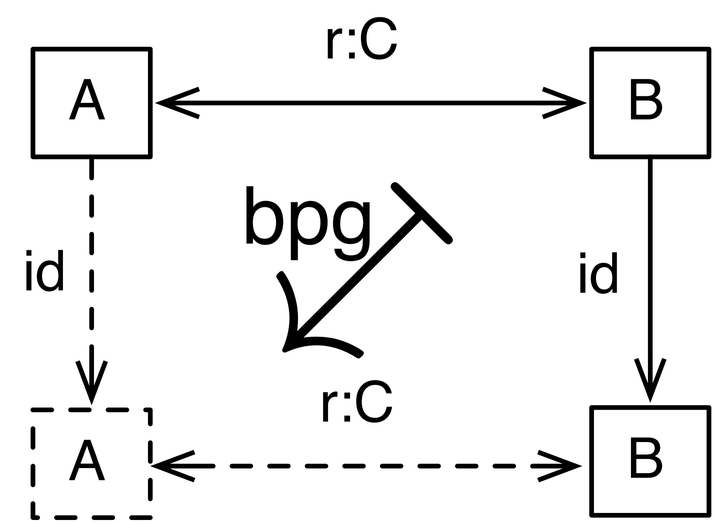
easy: just apply TGG rules wherever they match (typically quite efficient)

but: requires backtracking in general, so most TGG tools pose some (rather technical) restrictions

Proving stability

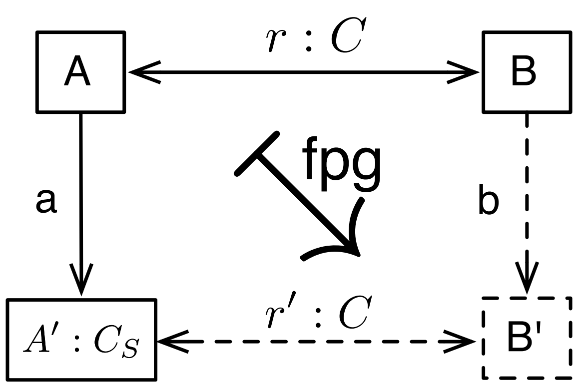
a TGG tool that actually inspects the delta to be propagated is trivially stable

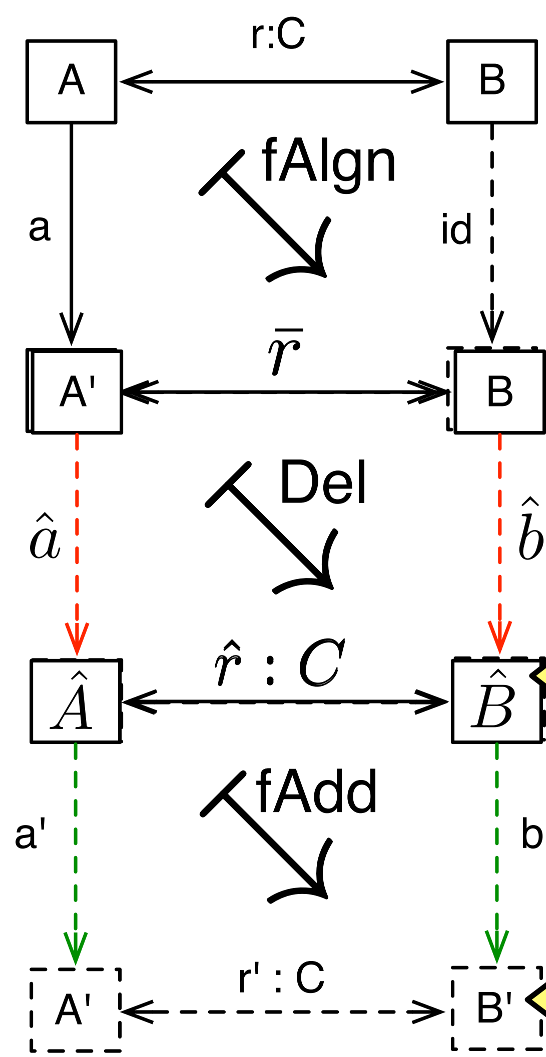




so **incremental** TGG tools are stable, **batch** TGG tools are not

Proving correctness

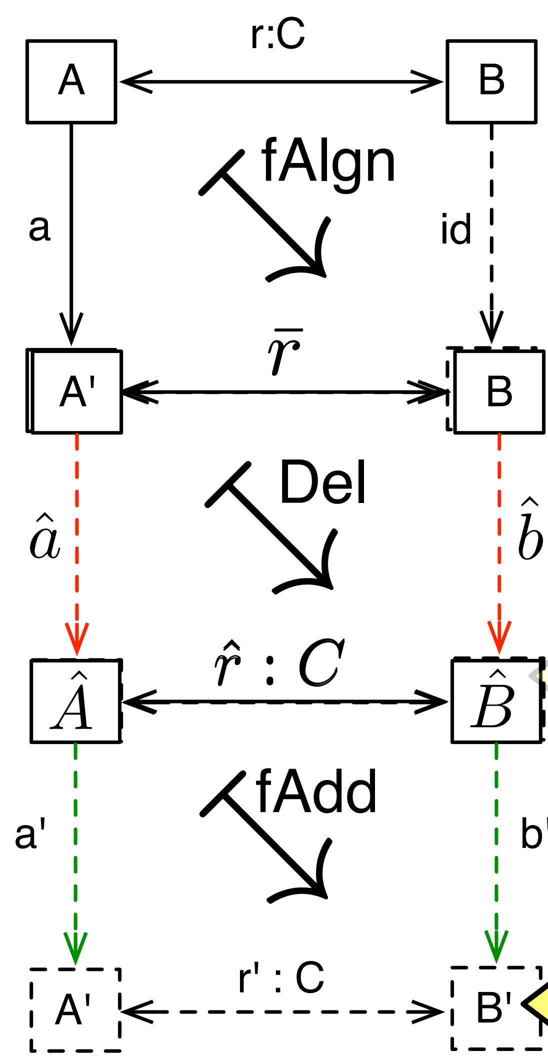




hard: show that **Del** (whatever strategy is applied) always produces a consistent intermediate result

easy: in each step, a TGG rule is applied, so if translation succeeds, the result is consistent by definition

Proving completeness

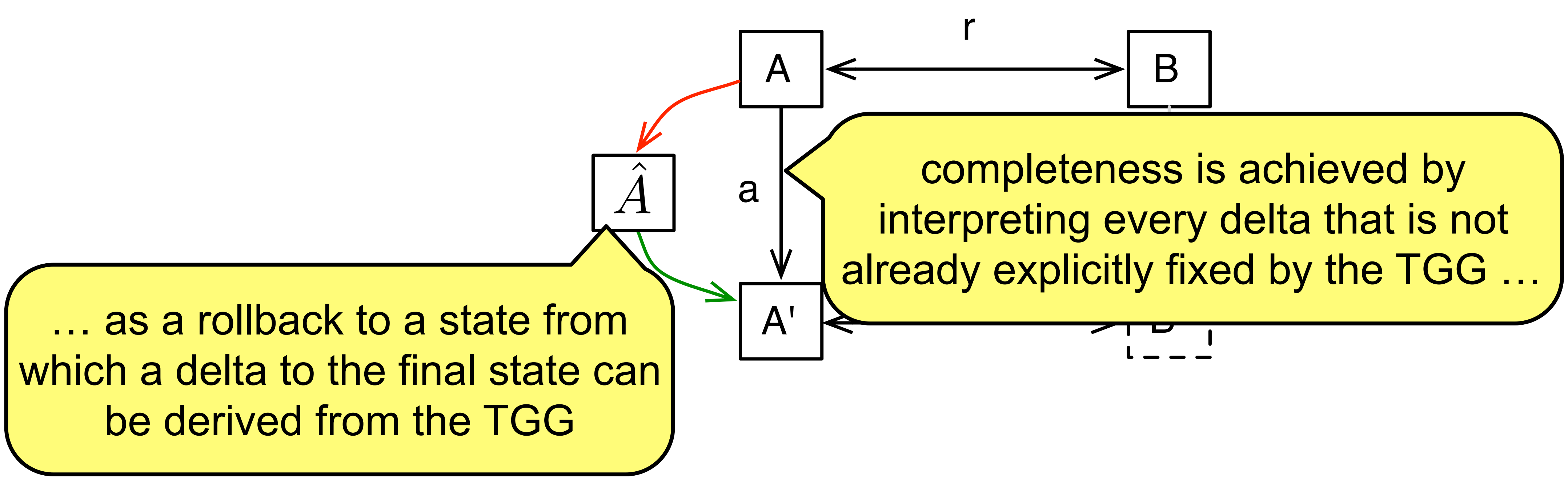


hard: show that **Del** (whatever strategy is applied) always produces a consistent intermediate result

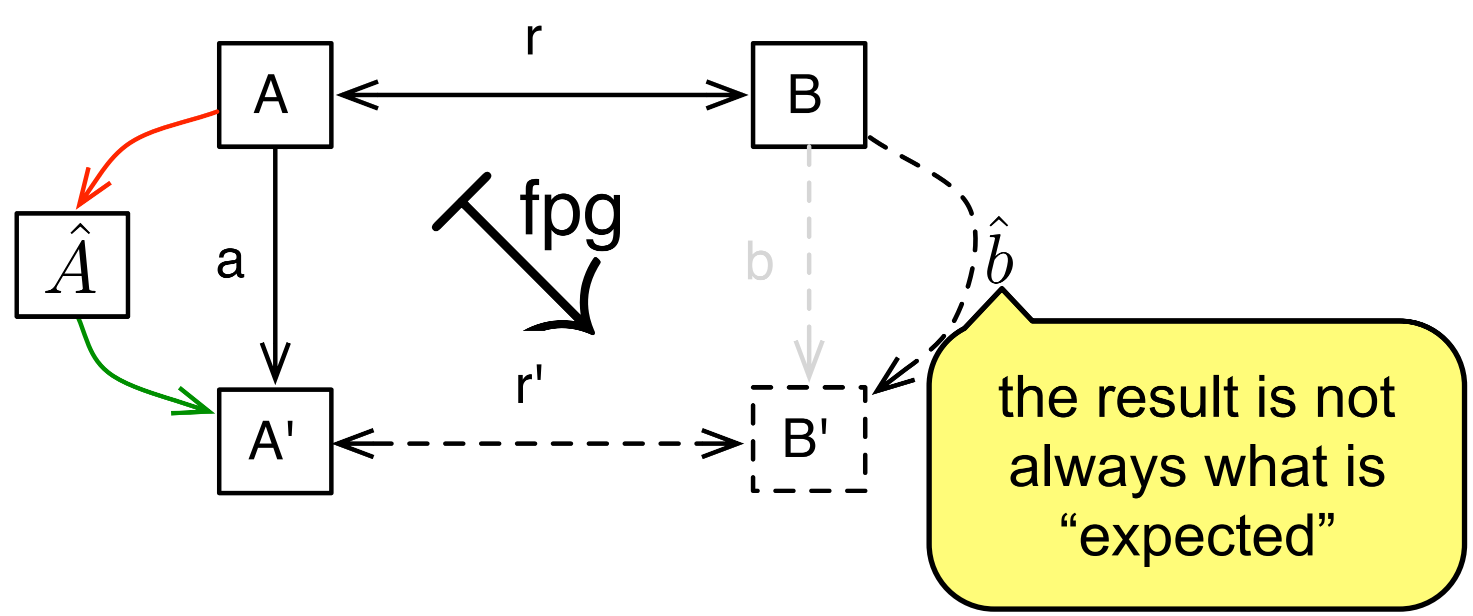
easy: if backtracking strategy is taken (but very inefficient)

hard: show that translation process succeeds without backtracking

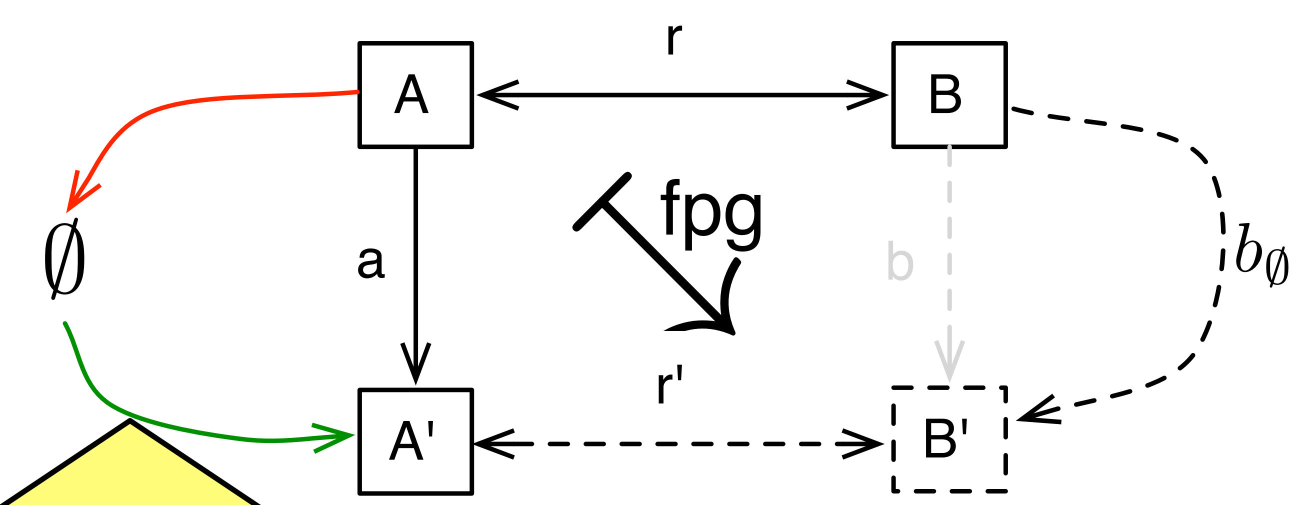
Some closing remarks on TGGs and "least change"



Some closing remarks on TGGs and “least change”



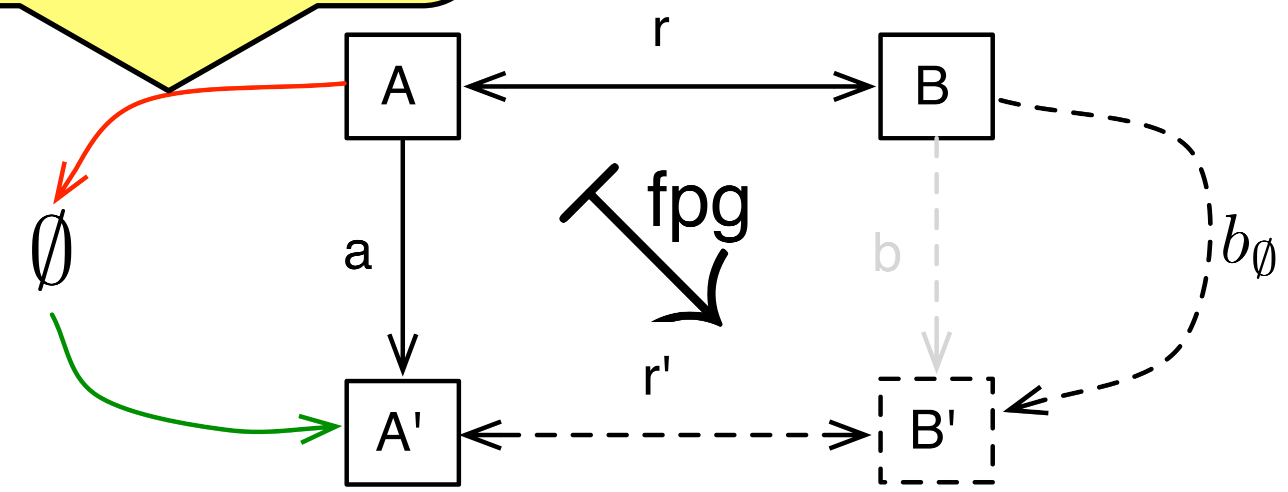
Some closing remarks on TGGs and “least change”



in the worst case, this can result in a complete rollback and re-translation (just as bad as batch, and less efficient!)

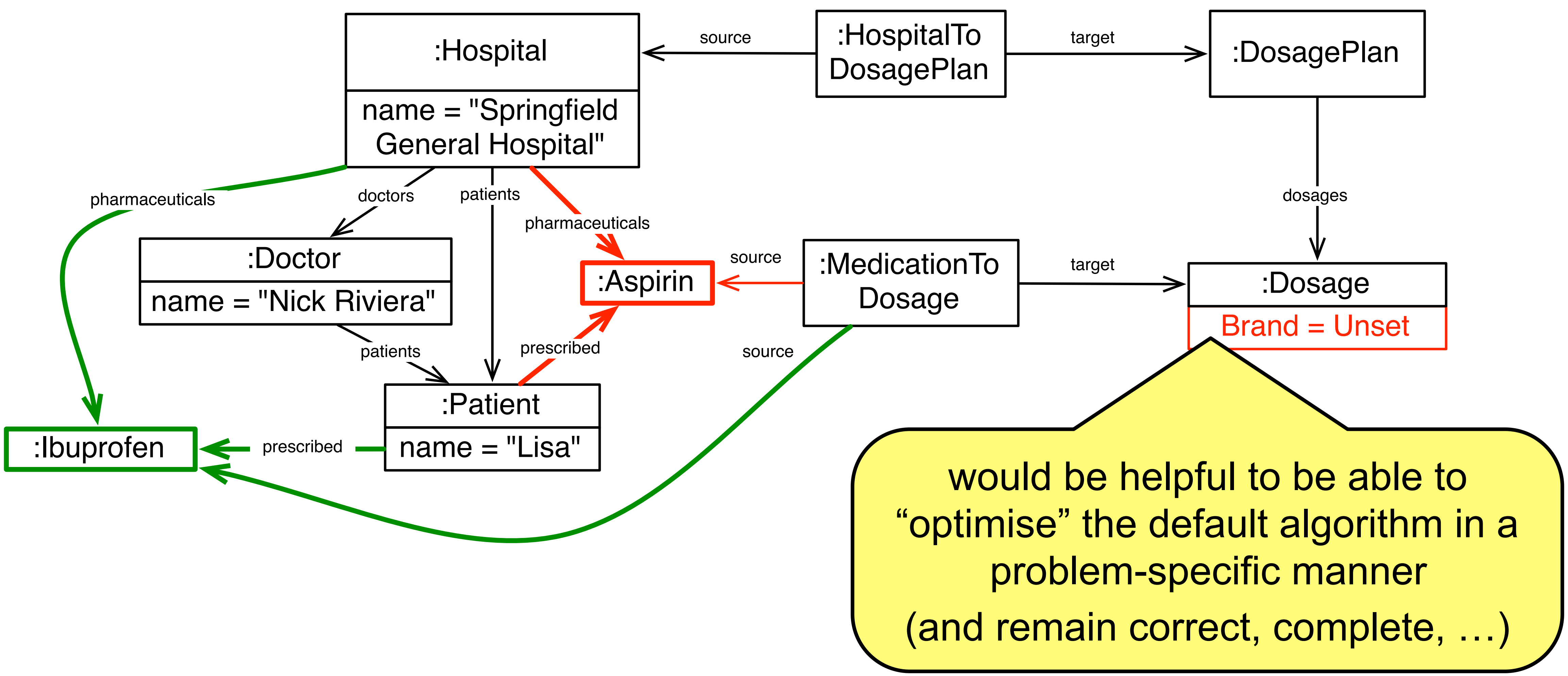
Some closing remarks on TGGs and “least change”

having this as a last resort (fallback) is often nice …

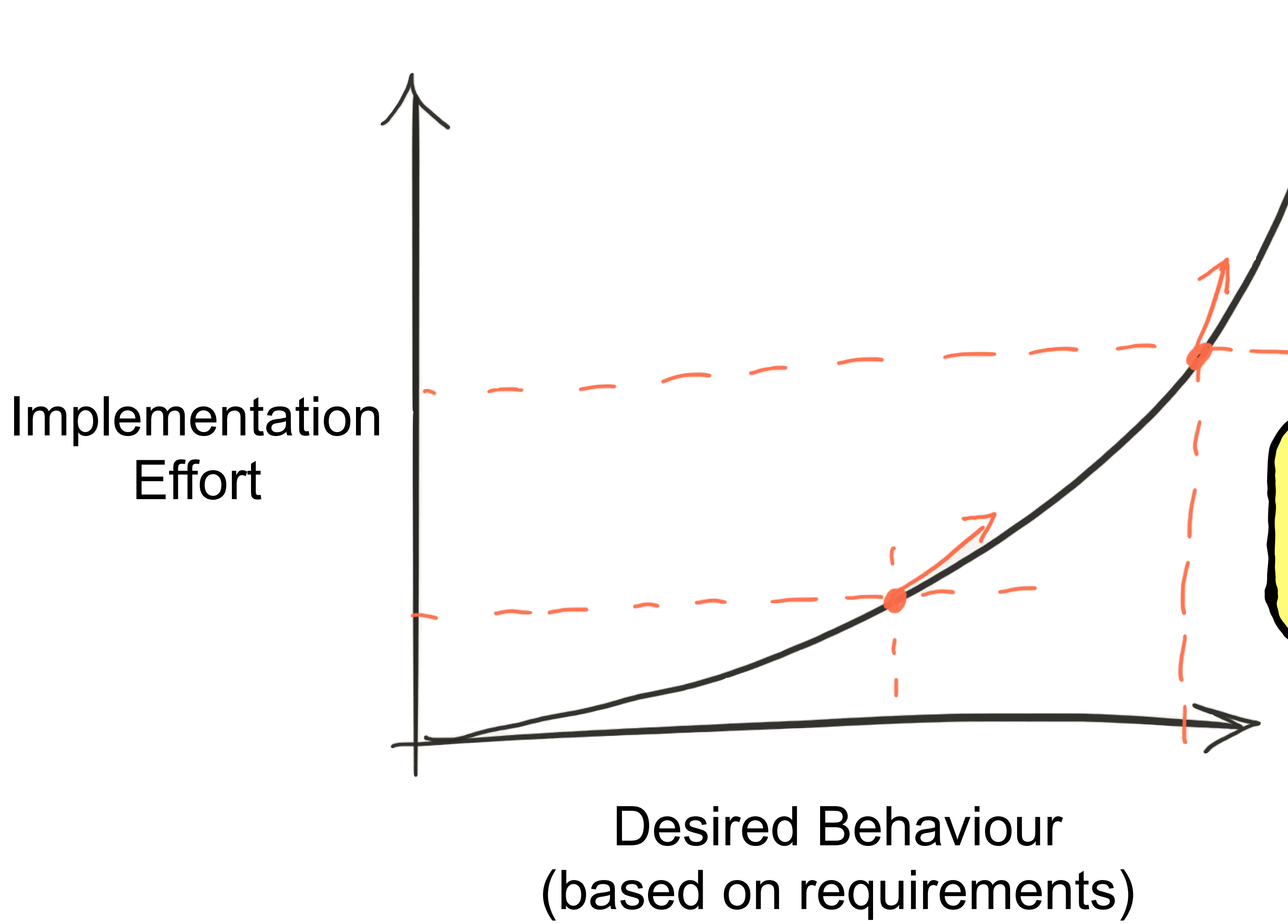


… but there is currently no elegant way to exercise fine granular control over the behaviour of TGG-based synchronisers

Running Example: Least Change?



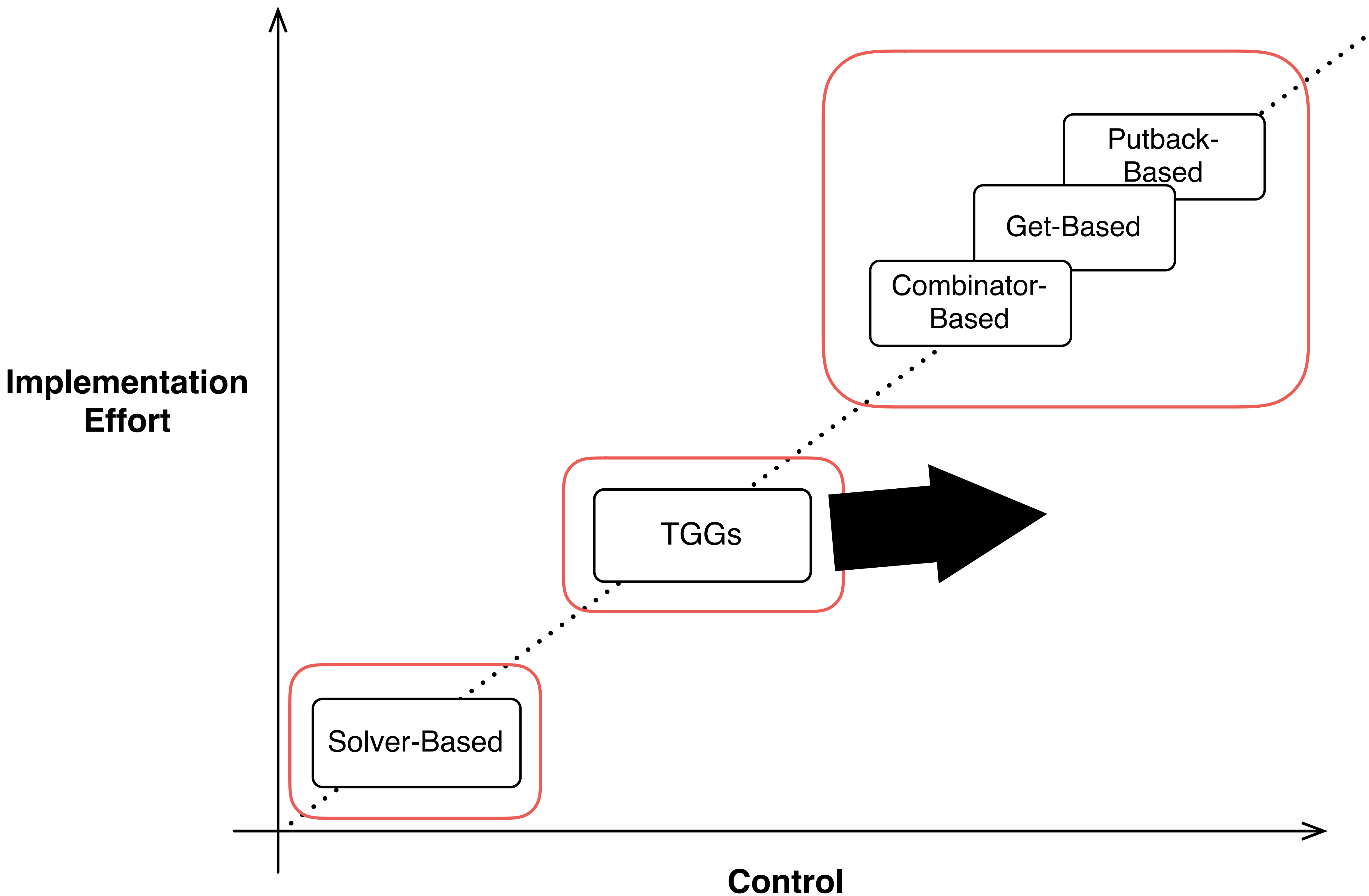
TGG Research Challenge (one of many! see [1])



TGG-based synchroniser development is not “smooth” :)

[1] 20 Years of Triple Graph Grammars: A Roadmap for Future Research. A Anjorin, E Leblebici, A Schürr - ECEASST, 2016

TGG Research Challenge (one of many!)



Task 3: A suite of examples with TGGs

1. Install VirtualBox from www.virtualbox.org
2. Download this VM: https://db.tt/gYgQMShZ
3. Open VM, start Eclipse with shortcut on desktop
4. Choose `workspaces/task3` as your workspace
5. Follow instructions from https://db.tt/GJ8fyeVm