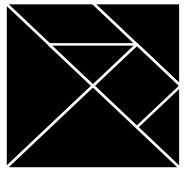
BX in Software Engineering

R. Paige, Z. Diskin and T. Maibaum

Dept of Computer Science, University of York

Computing & Software, McMaster University

GSDLab, University of Waterloo



Contents

- Foundations:
  - Terminology: model, metamodel, types of transformations, traceability.
  - Typical applications
- Conceptual and theoretical overview
- SE tools for BX
- Integration questions

Model

- A structured description of phenomena of interest.
  - Processed by automated tools.
- Structure can be provided in a number of ways:
  - Schema (explicit or implicit), typing rules, constraints, metamodel…
  - Normally graphs.
  - Don’t restrict ourselves to ‘just’ EMF/Ecore.

Metamodel

- A metamodel is a model of the abstract syntax and (parts of the) static semantics of a language.
- The relationship between model and metamodel is called *conformance.*
  - Namely, a model conforms to a metamodel.
- Techniques for metamodelling: EMF, MOF, XML schemas, typed graphs, …
  - NB: metamodel != grammar

Constraints

- Metamodels capture some static semantic constraints (like multiplicity).
- Richer constraints may be needed to prevent undesirable/illegitimate models from being instantiated.
- If multiple models/metamodels are being used, inter-model constraints may be used to establish *consistency.*
  - e.g., EVL, xlinkit, OCL (with a union metamodel)
  - e.g., QVT-R checkonly mode

Operations

- Given metamodels and conforming models, we may want to apply operations to them.
  - Match/compare
  - Merge
  - Check (constraint, critique)
  - Generate (text, concrete syntax)
  - Migrate
  - Transform (update-in-place, source-to-target, …)
  - ….

Transformations

- *Unidirectional*: from a source model to a target model.
  - Defined in terms of metamodels, usually languages are “linguistically similar”.
- *Update-in-place*: modifications made to one source/target model; normally unidirectional.
- *Bidirectional*: source and target models are established to be “consistent” at well-defined points in time.
  - e.g., after repository check-in; after check-then-enforce has run in QVT-R

Traceability

- All operations on models have side-effects: they generate trace-links.
- Trace-links relate model elements (not just models).
  - Different types of trace-links (e.g., *contains*, *regenerates*, - see Aizenbud-Reshef's work)
- Most MDD tools (Epsilon, ATL) generate such trace-links and allow them to be persisted.
- Basis for validation and verification.

SE Scenarios

- Round-trip engineering (from models to code)?
- Supporting multiple stakeholders editing the same models?
- Synchronising documentation and code?
- Impact analysis for helping to maintain certification/assurance arguments?
- Reflecting analysis results in models?
- …

Contents

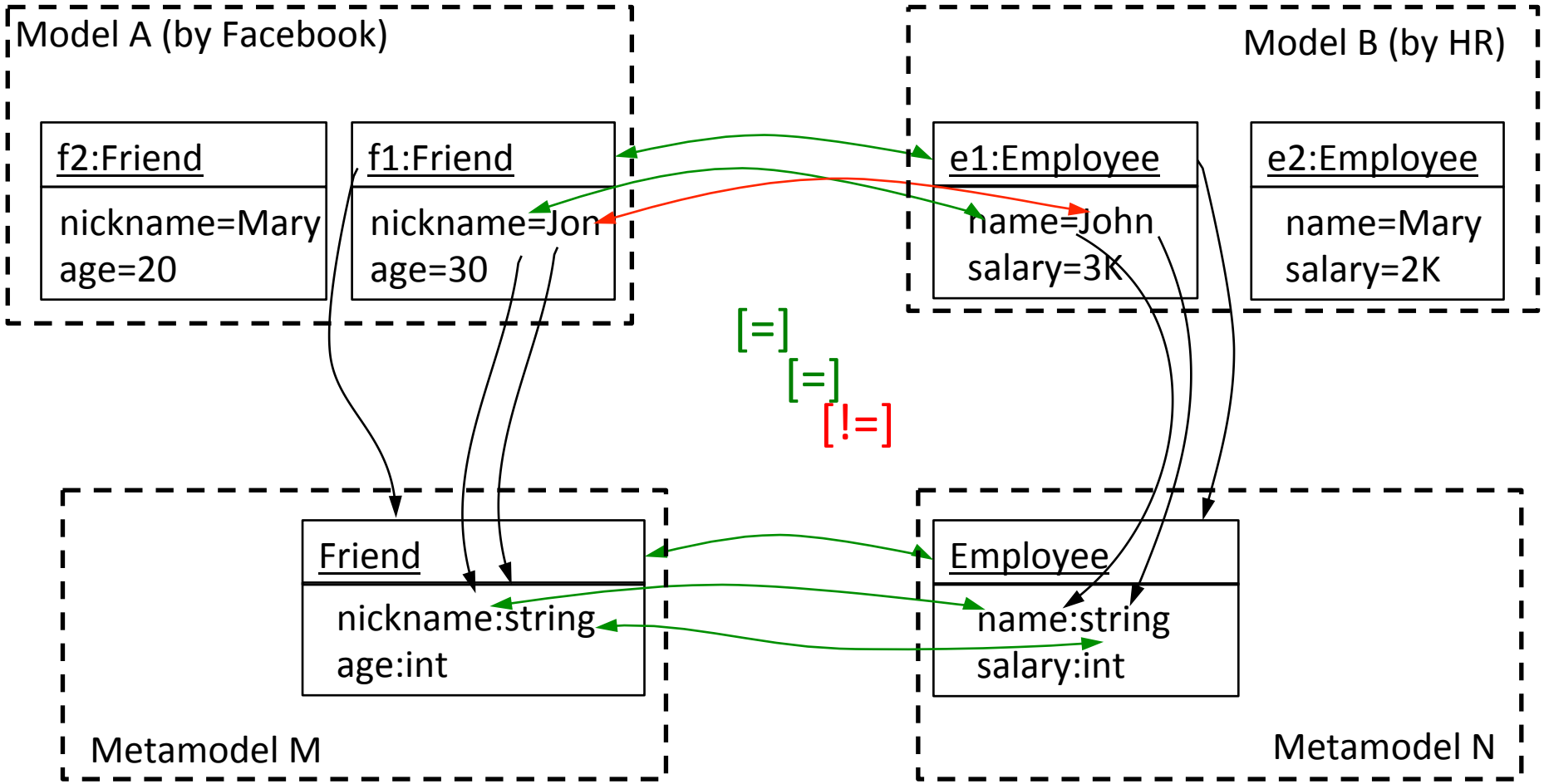
- Foundations:
  - Terminology: model, metamodel, types of transformations, traceability.
  - Typical applications
- Conceptual and theoretical overview
- SE tools for BX
- Integration questions

Contents

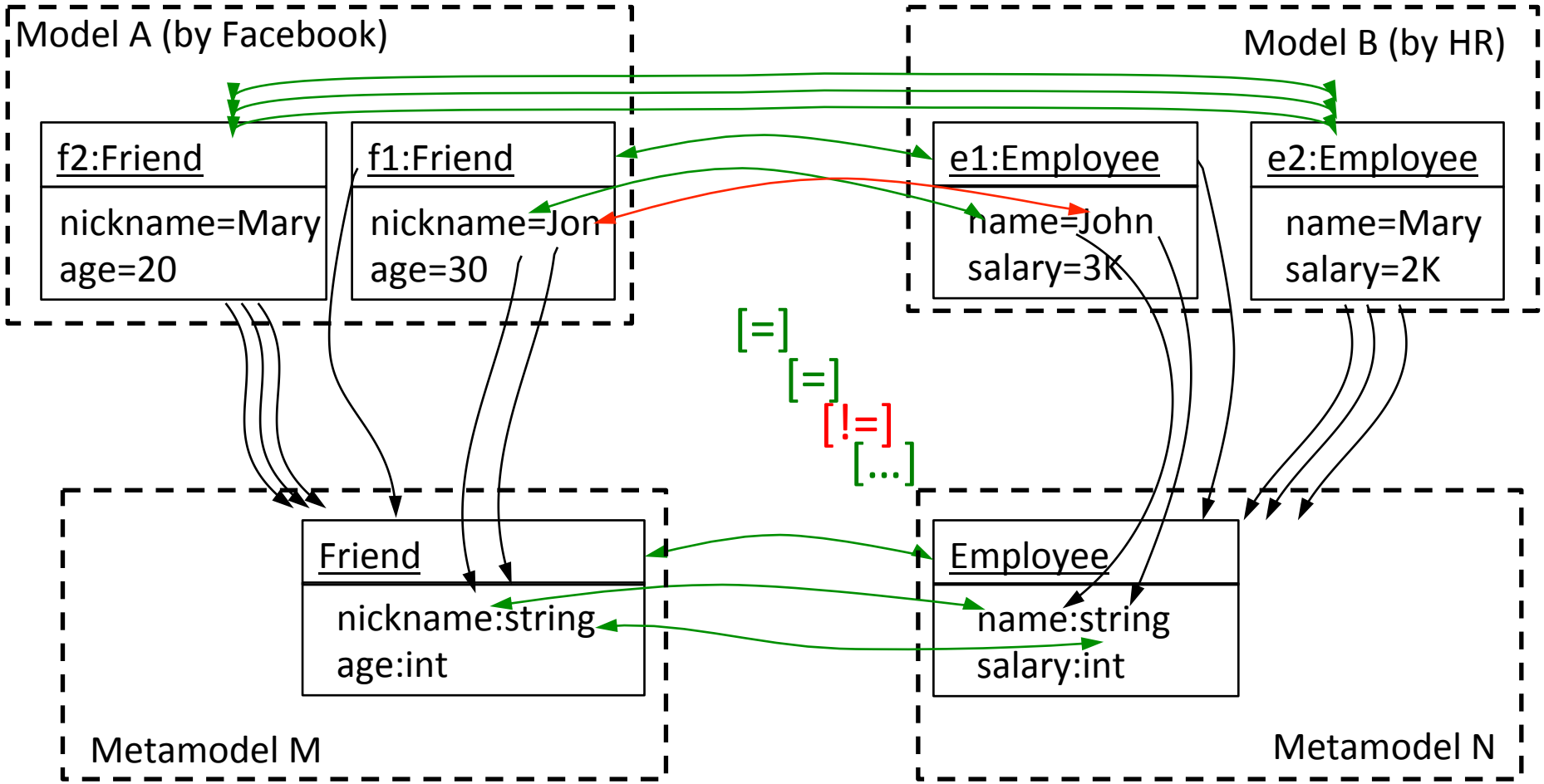
- Foundations:
- Conceptual and theoretical overview
  - Deltas
  - Delta composition
  - Delta propagation
  - Delta lenses (= Algebras for delta management)
- SE tools for BX
- Integration questions

Traceability management

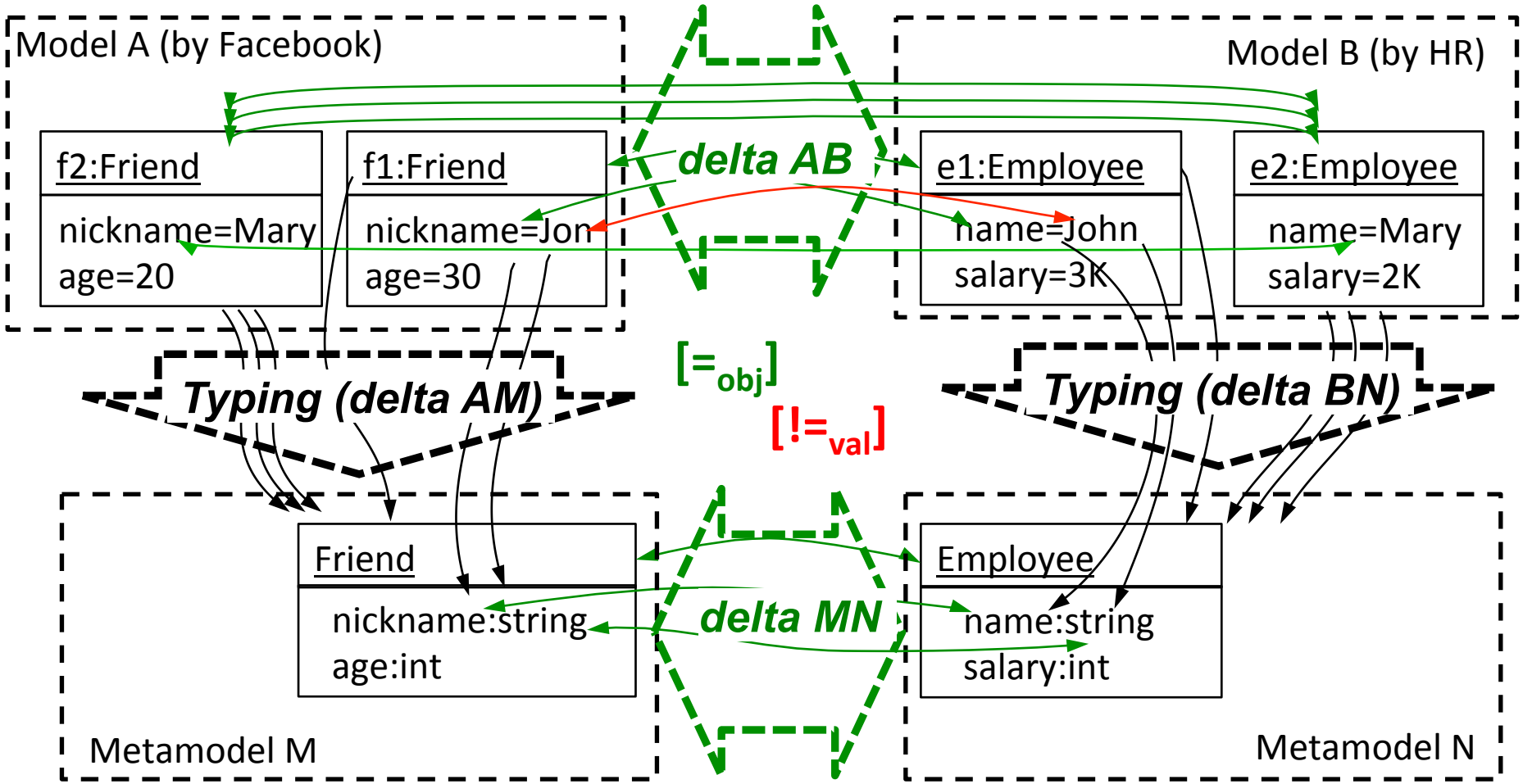
Deltas, or Trace models



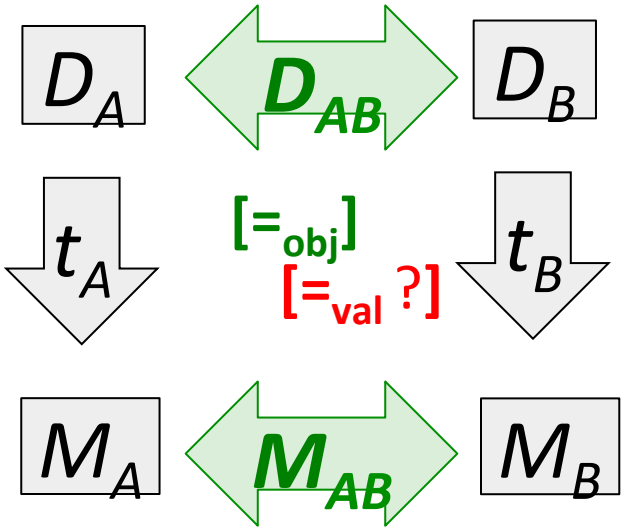
Deltas, or Trace models



Deltas, or Trace models



Deltas abstractly

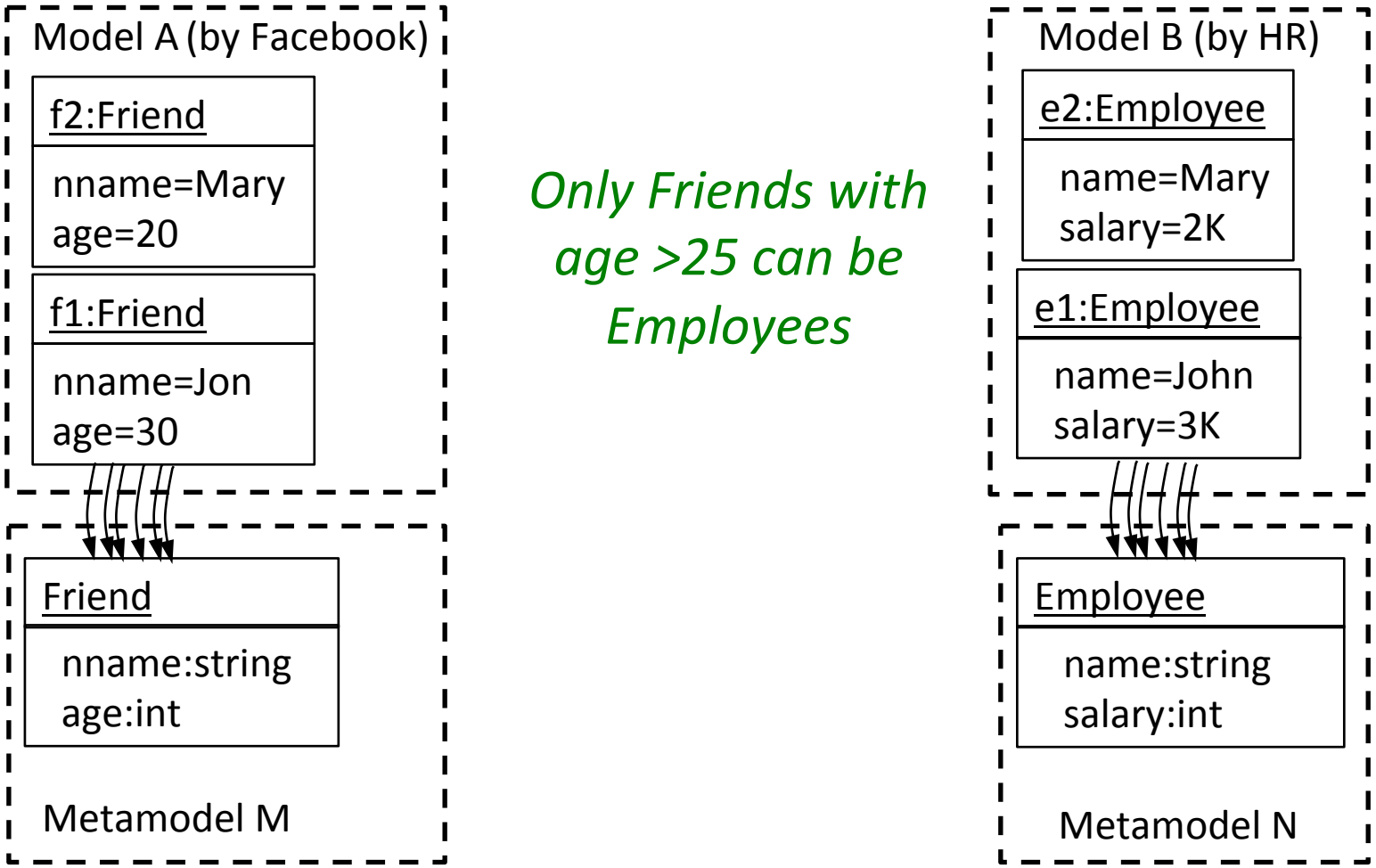


- Nodes are graphs
- Unidir arrows – graph mappings
- Bidir arrows – sets of reified bidir links (= spans of graph mappings)

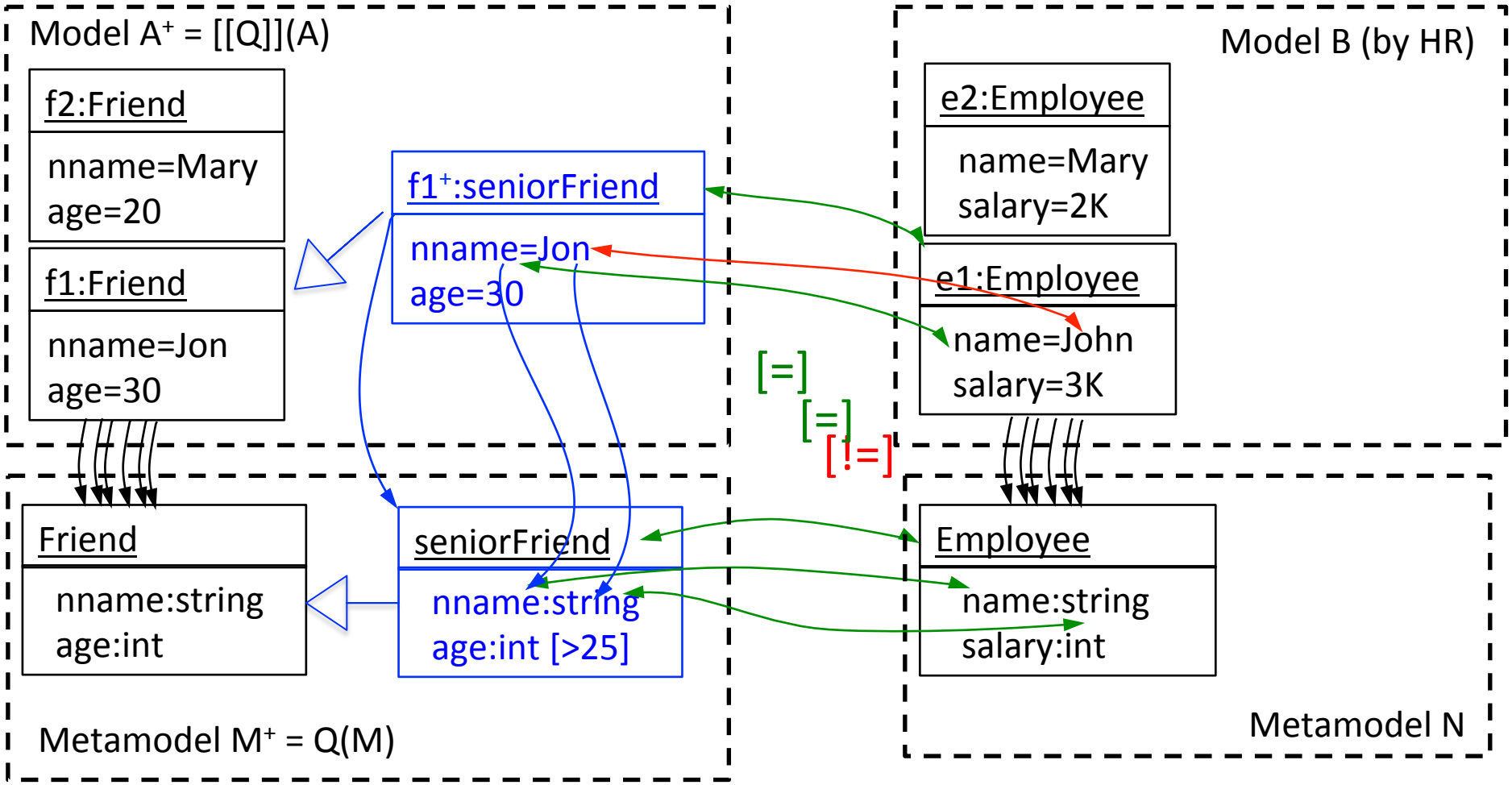
satisfy consistency conditions ?

- Nodes are graph mappings
- Bidir arrow – the green part of the diagram of graph mappings on the left

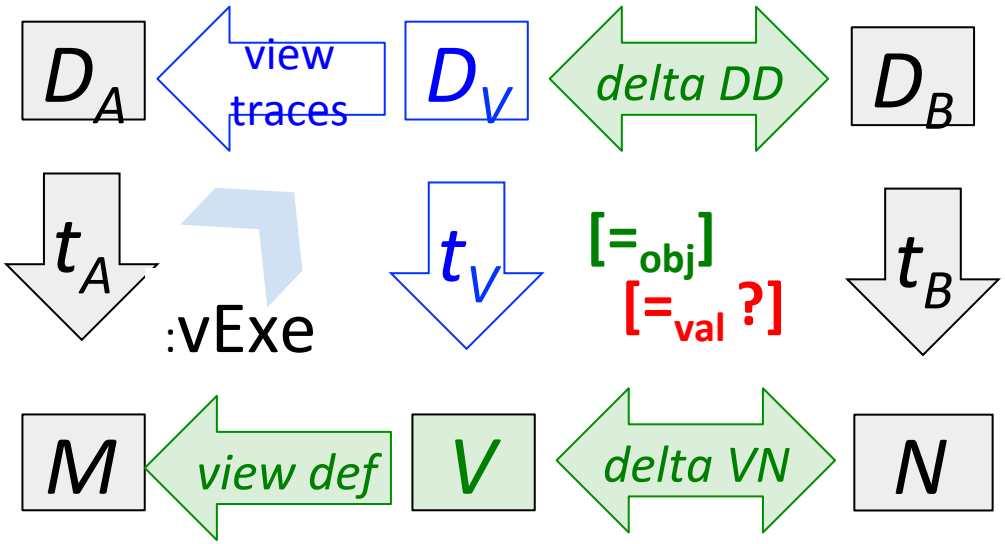
Complex deltas (with queries)

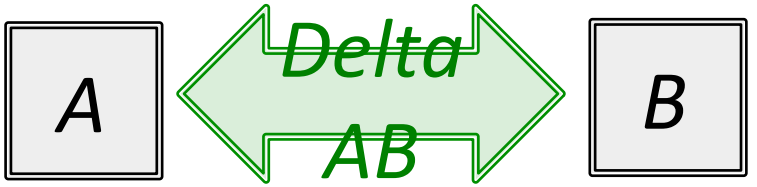


Complex deltas (with queries)



Complex deltas abstractly



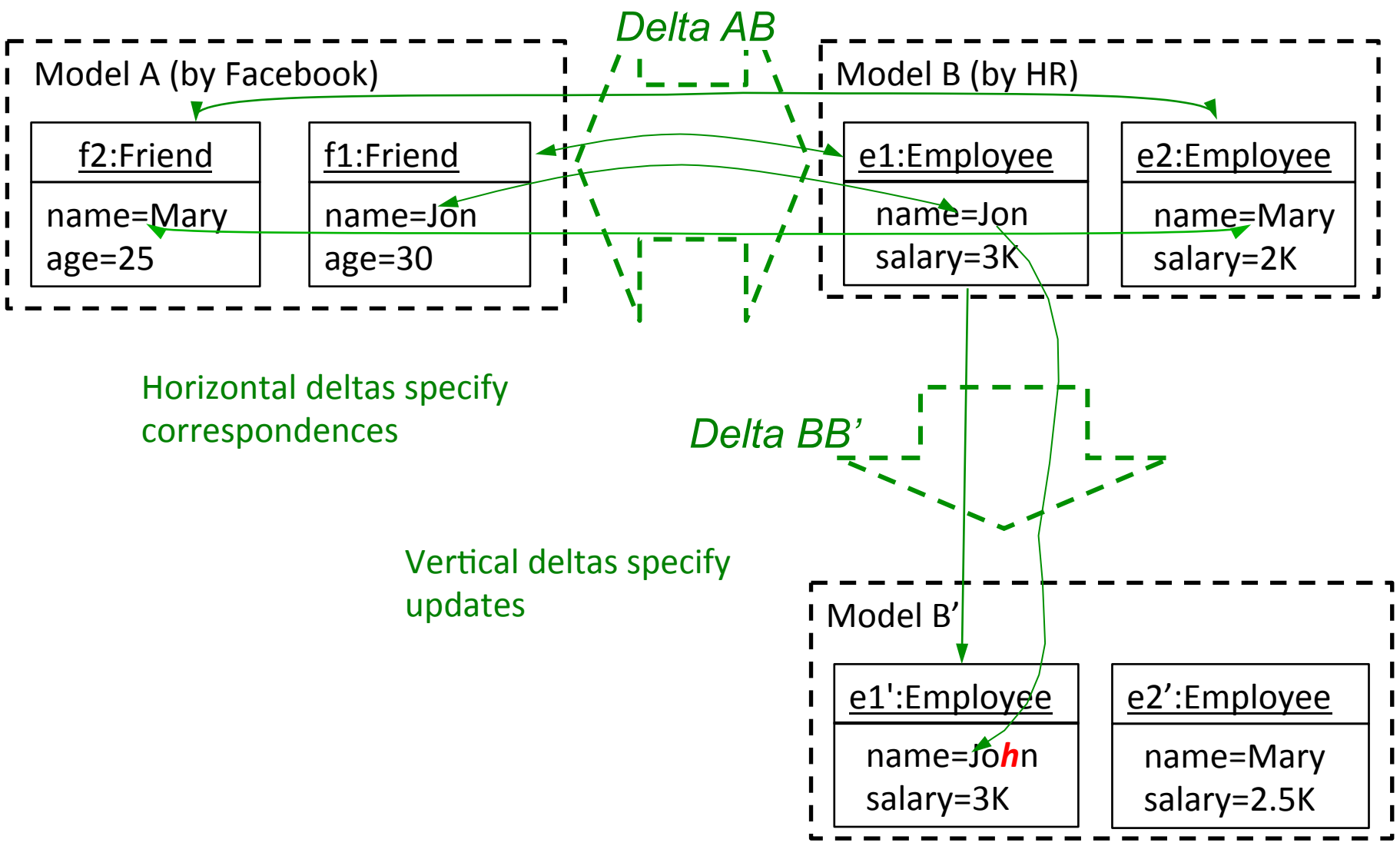


satisfy consistency conditions ?

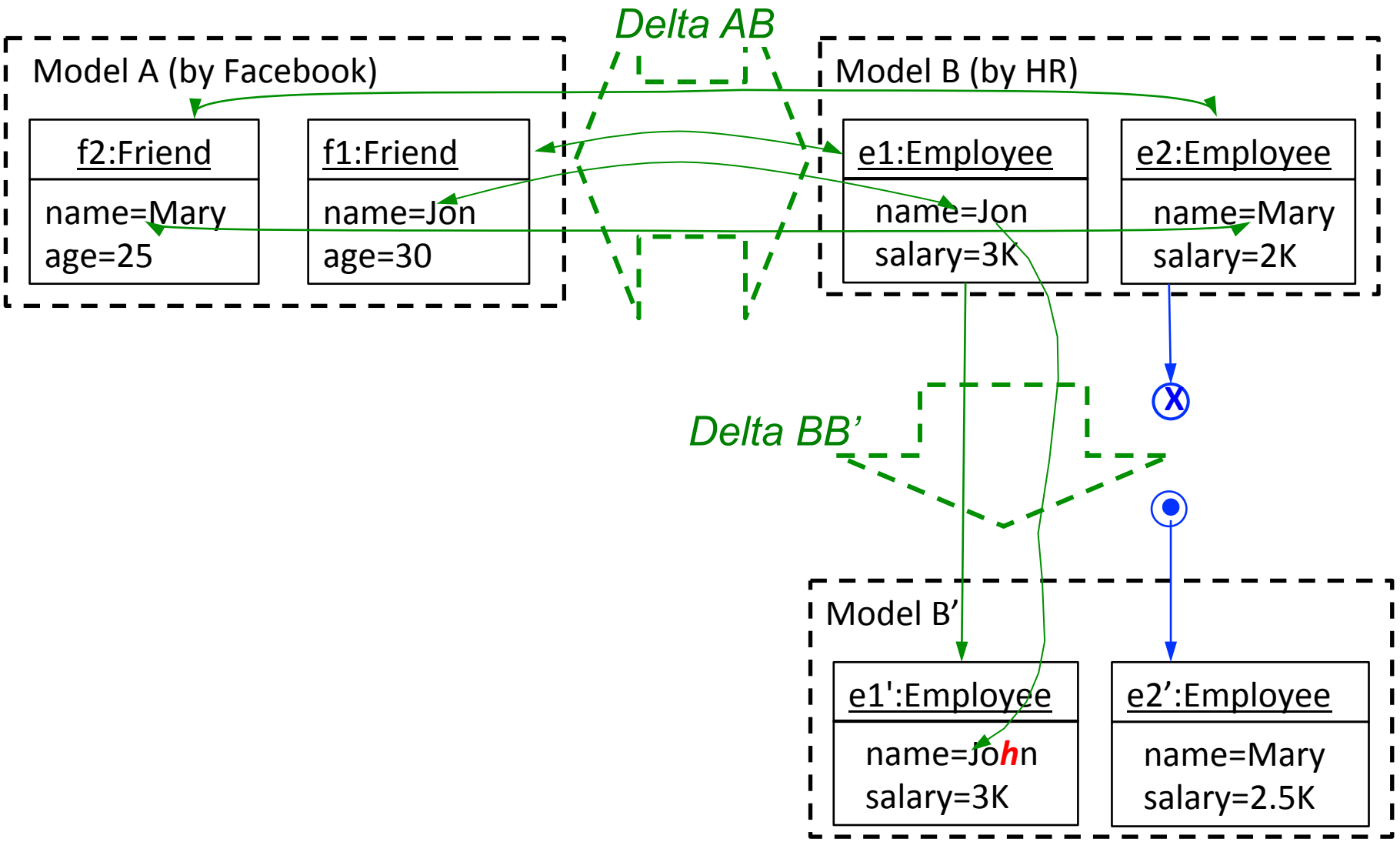
- Nodes are graphs
- Unidir arrows – graph mappings
- Bidir arrows – sets of reified bidir links (= spans of graph mappings)

- Nodes are graph mappings
- Bidir arrow – the diagram of graph mappings on the left

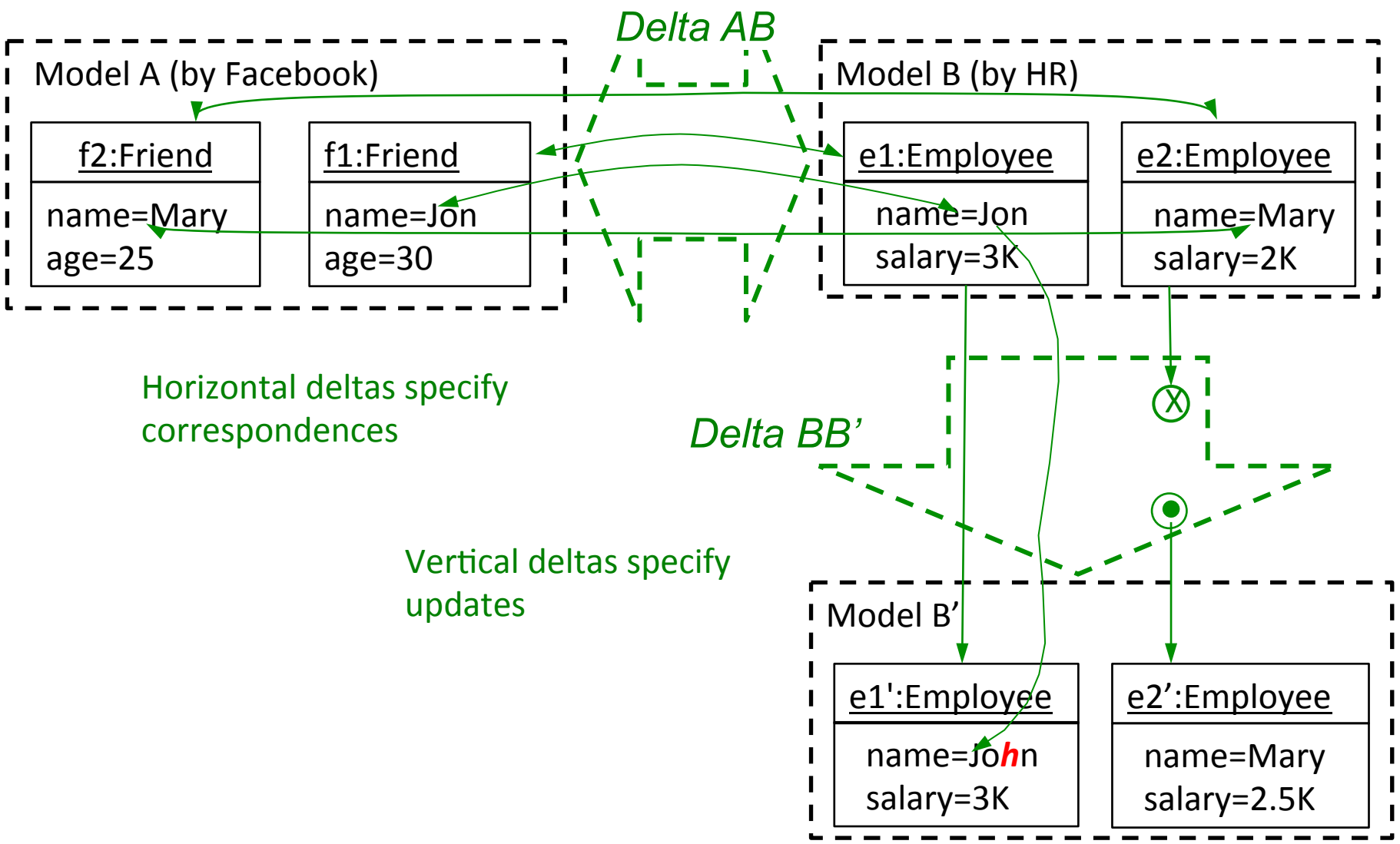
Updates as deltas



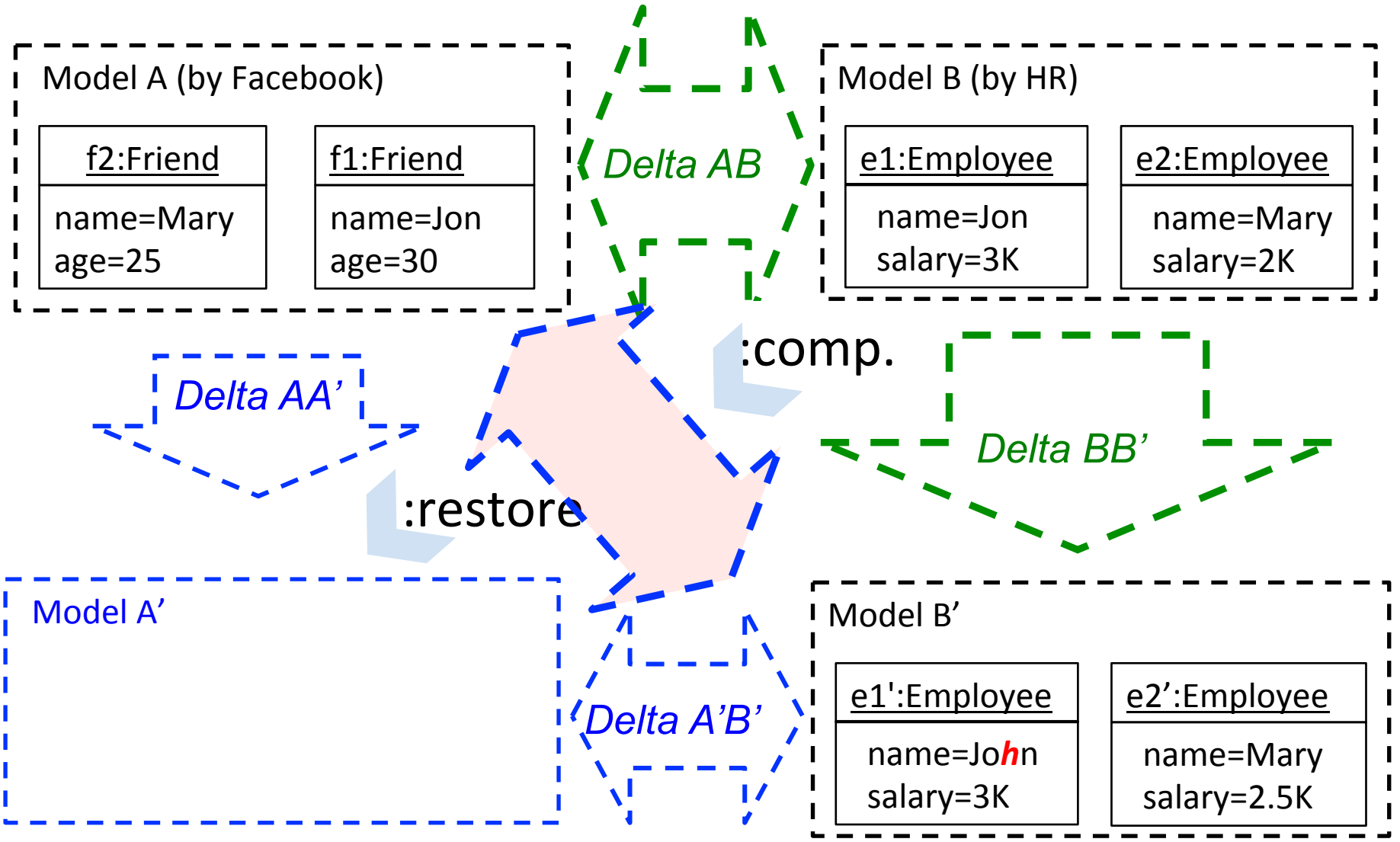
Updates as deltas



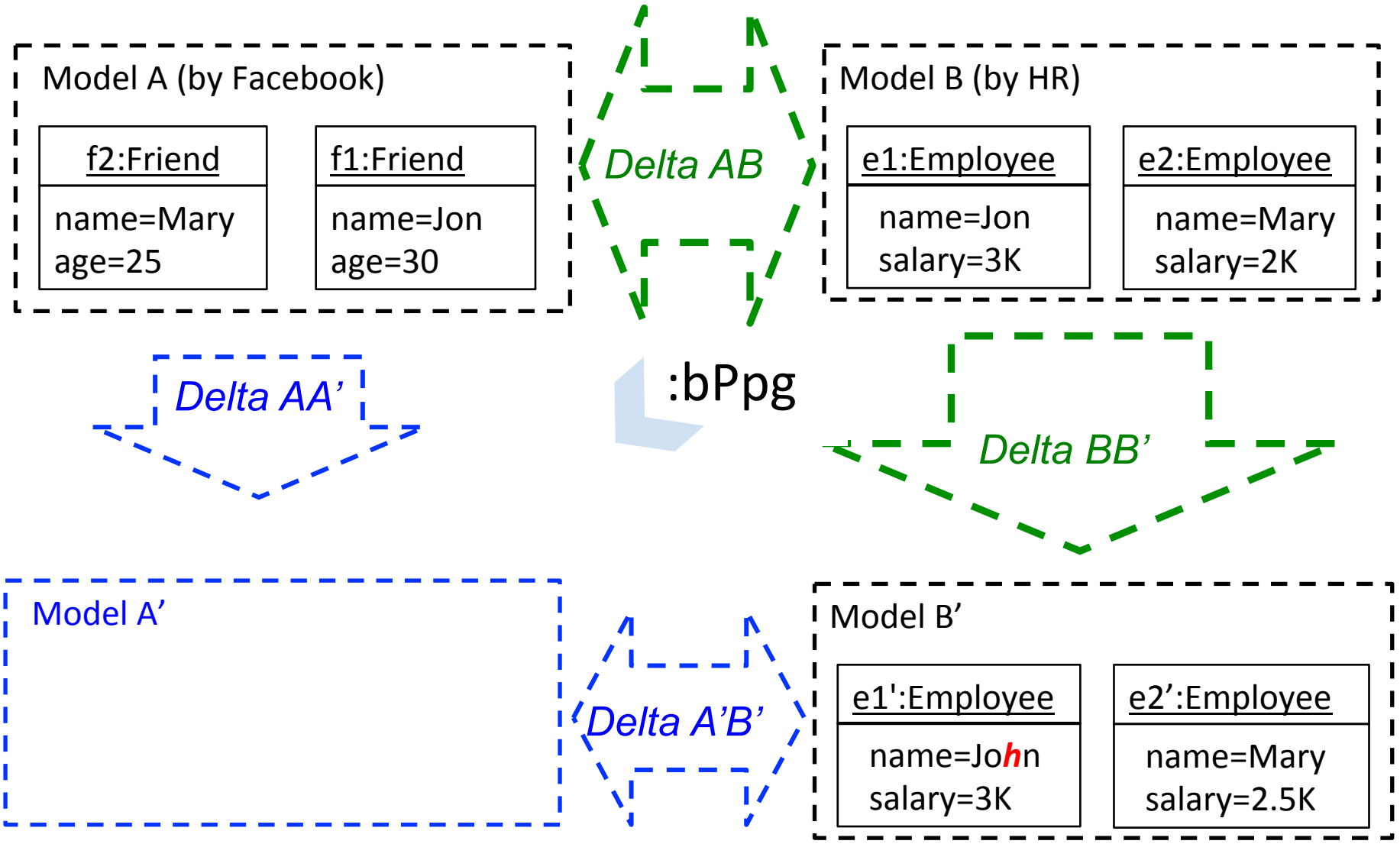
Updates as deltas



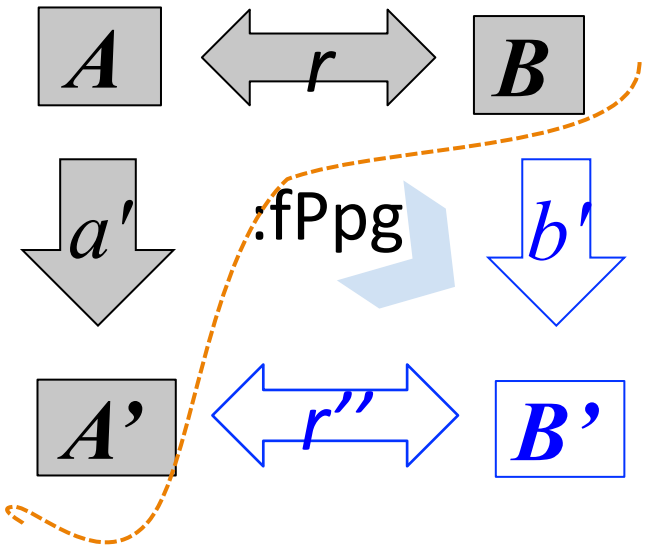
Consistency restoration



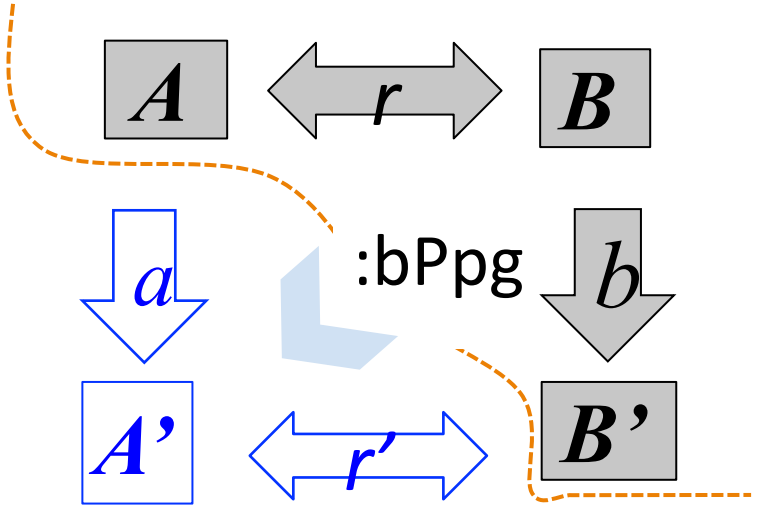
Delta/Update Propagation



Delta propagation abstractly

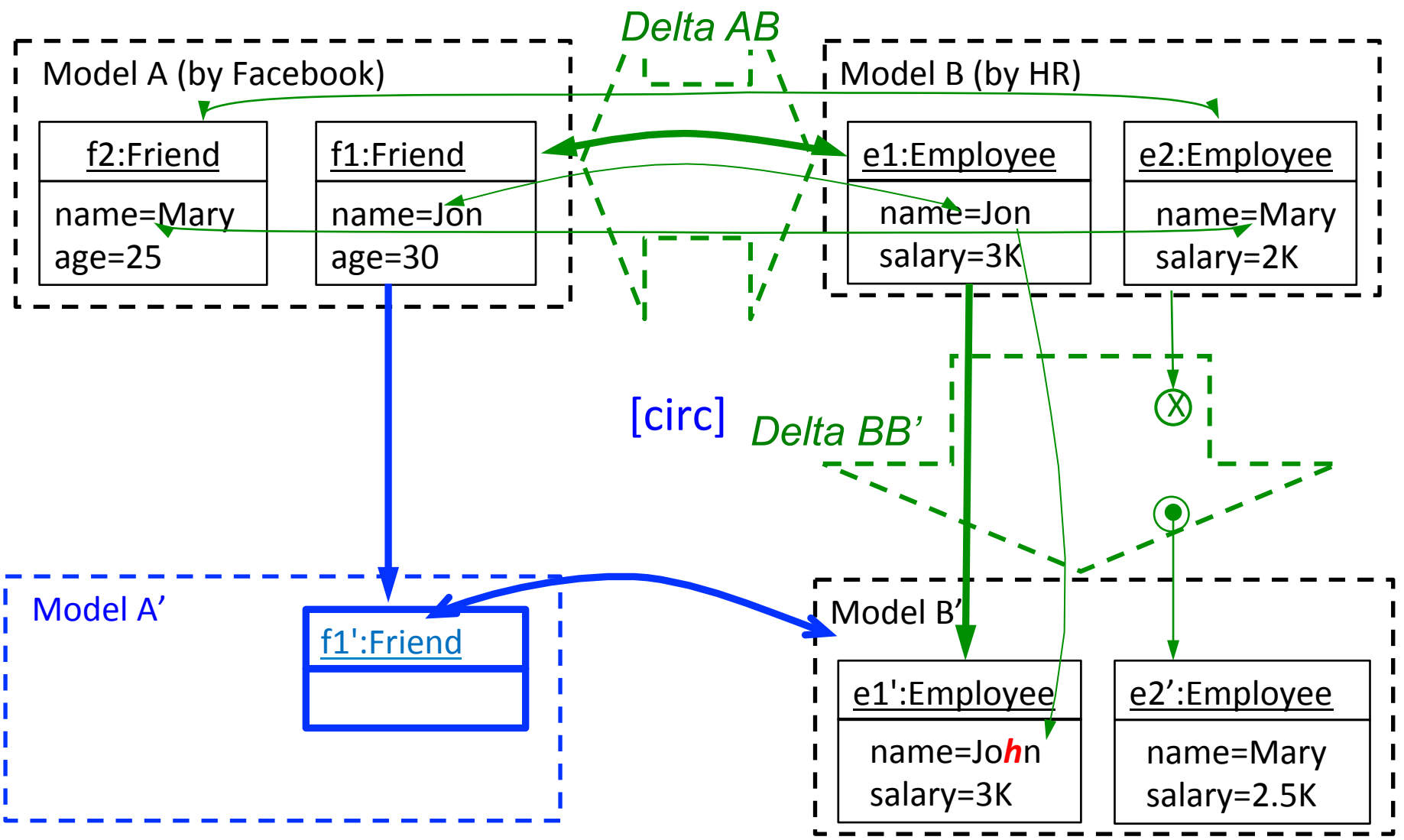


forward Propagation

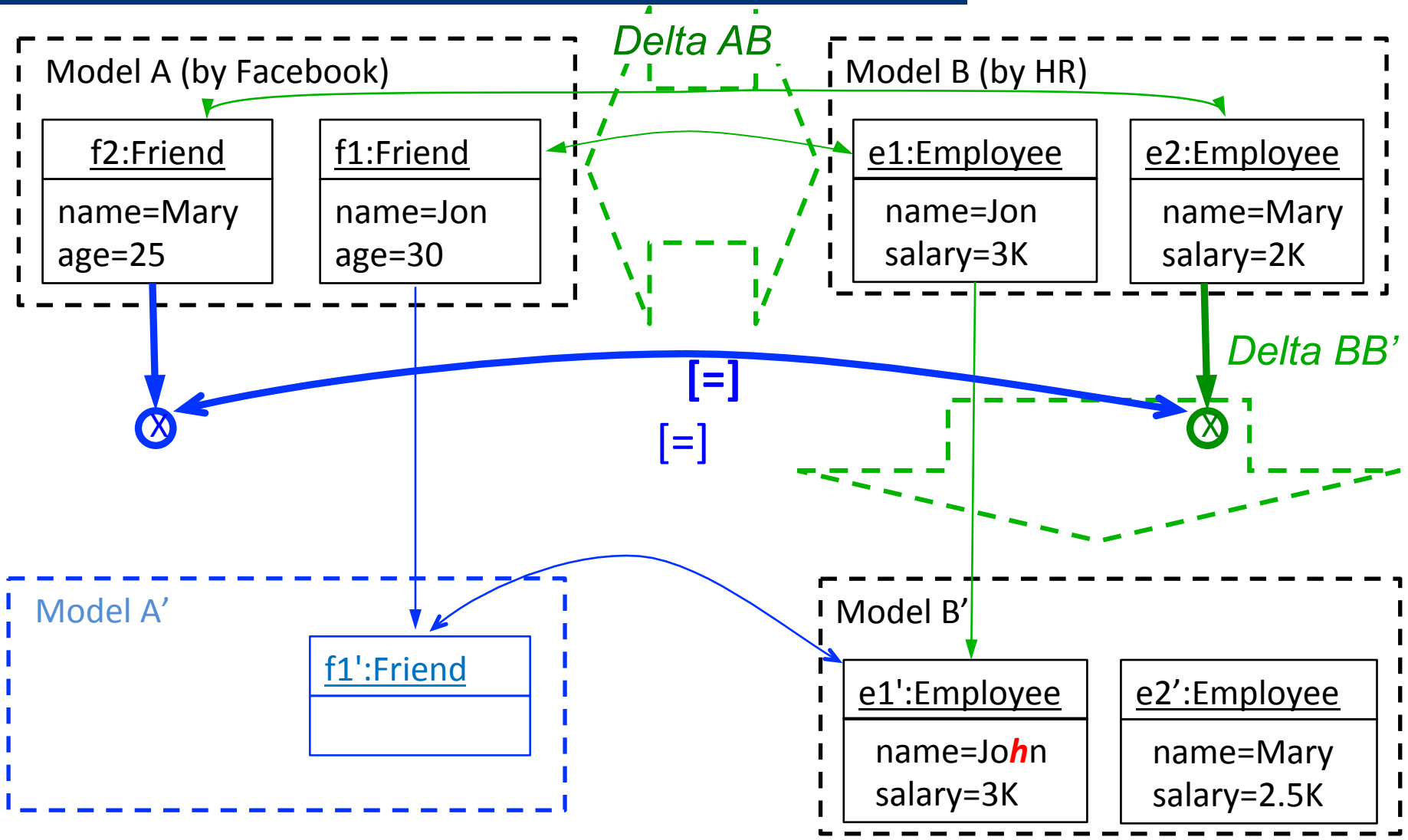


backward Propagation

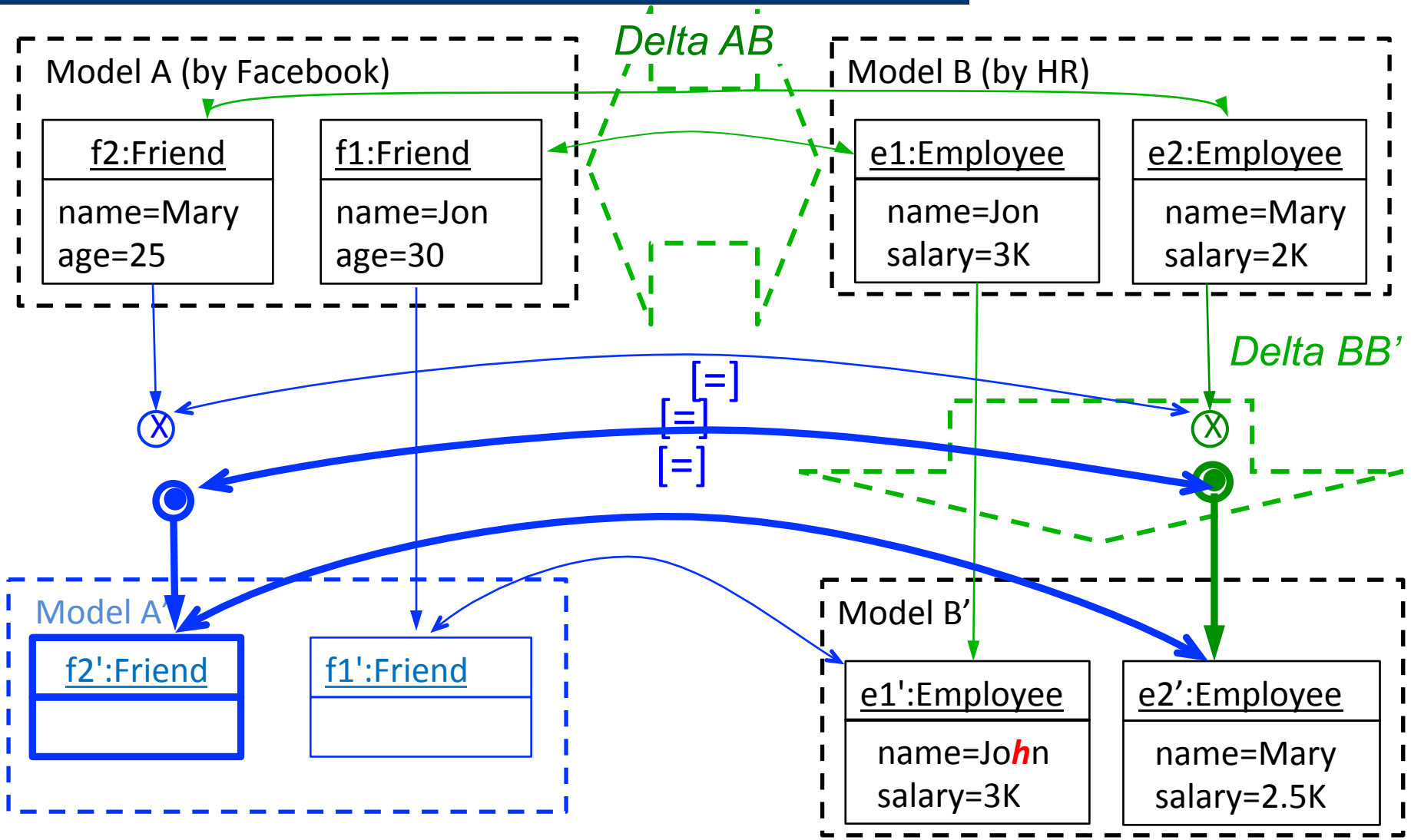
OID Propagation



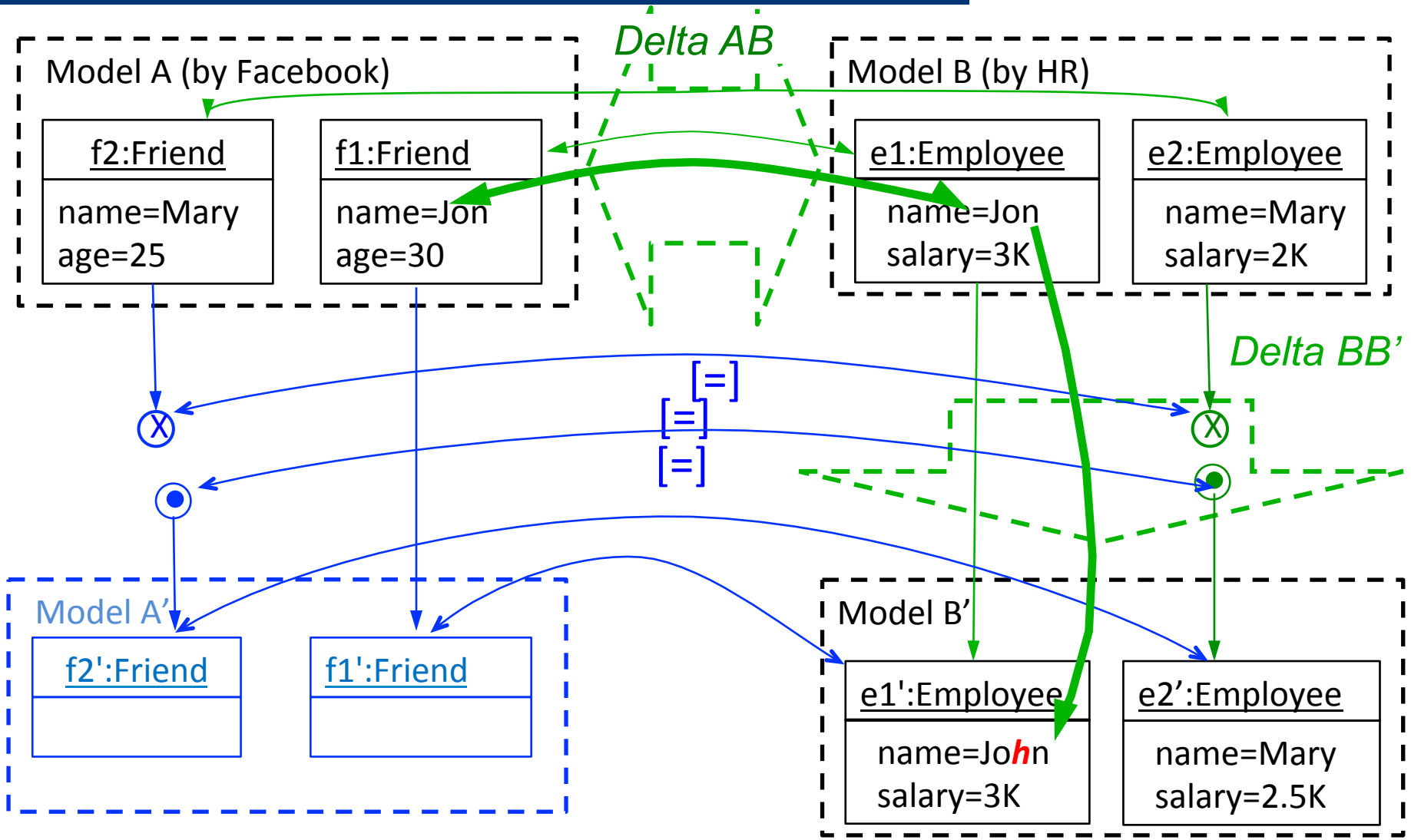
OID Propagation



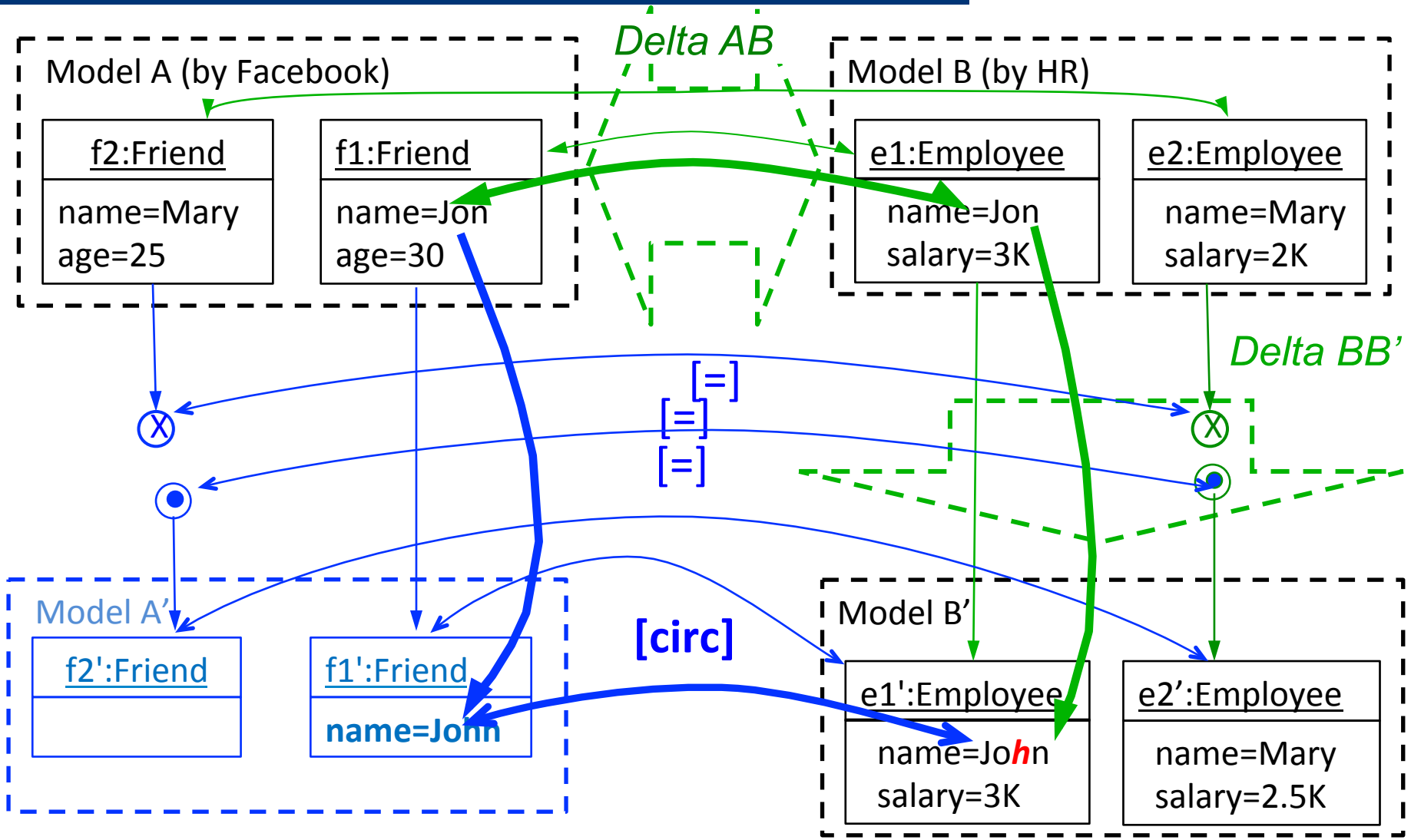
OID Propagation



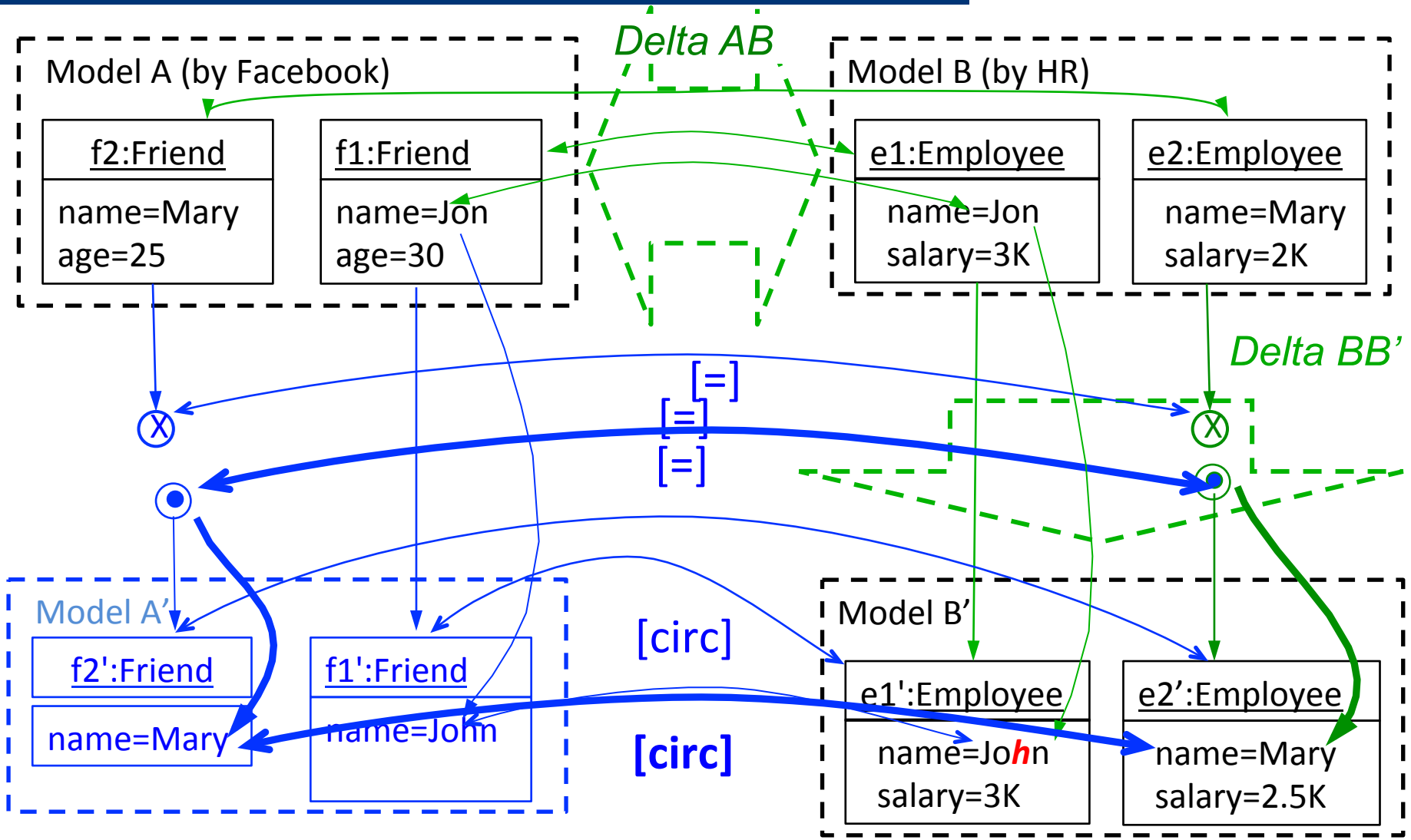
Attribute Propagation



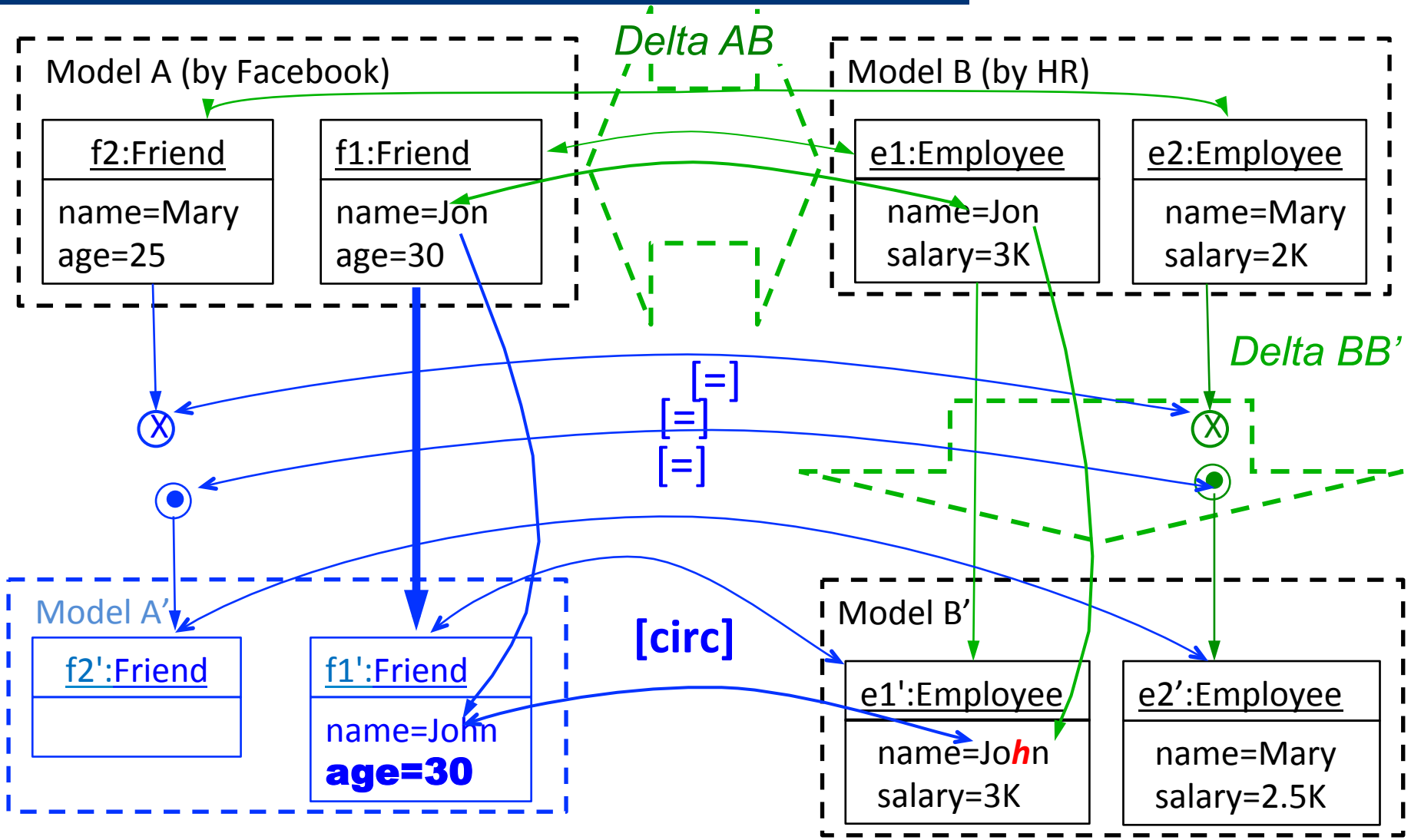
Attribute Propagation



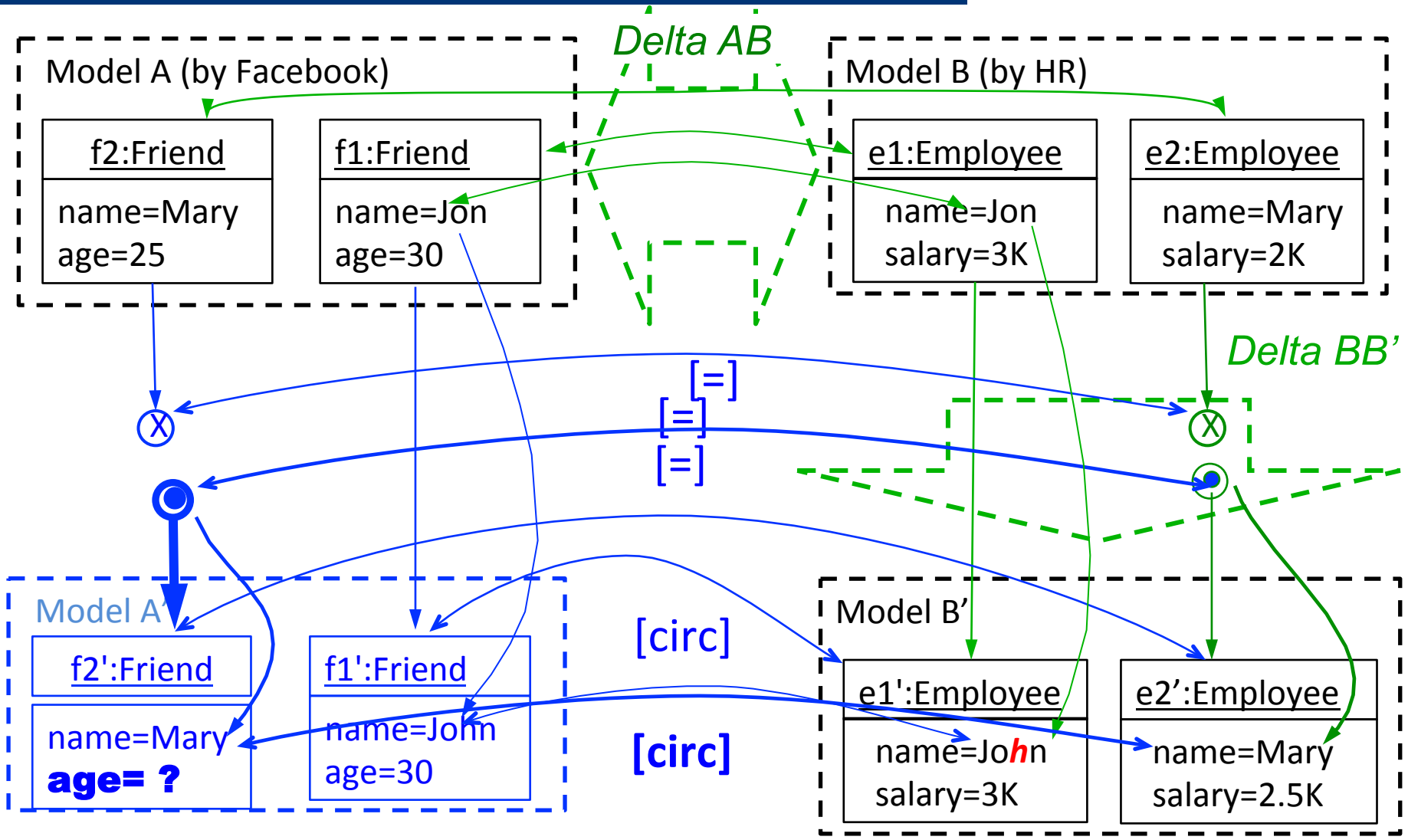
Attribute Propagation



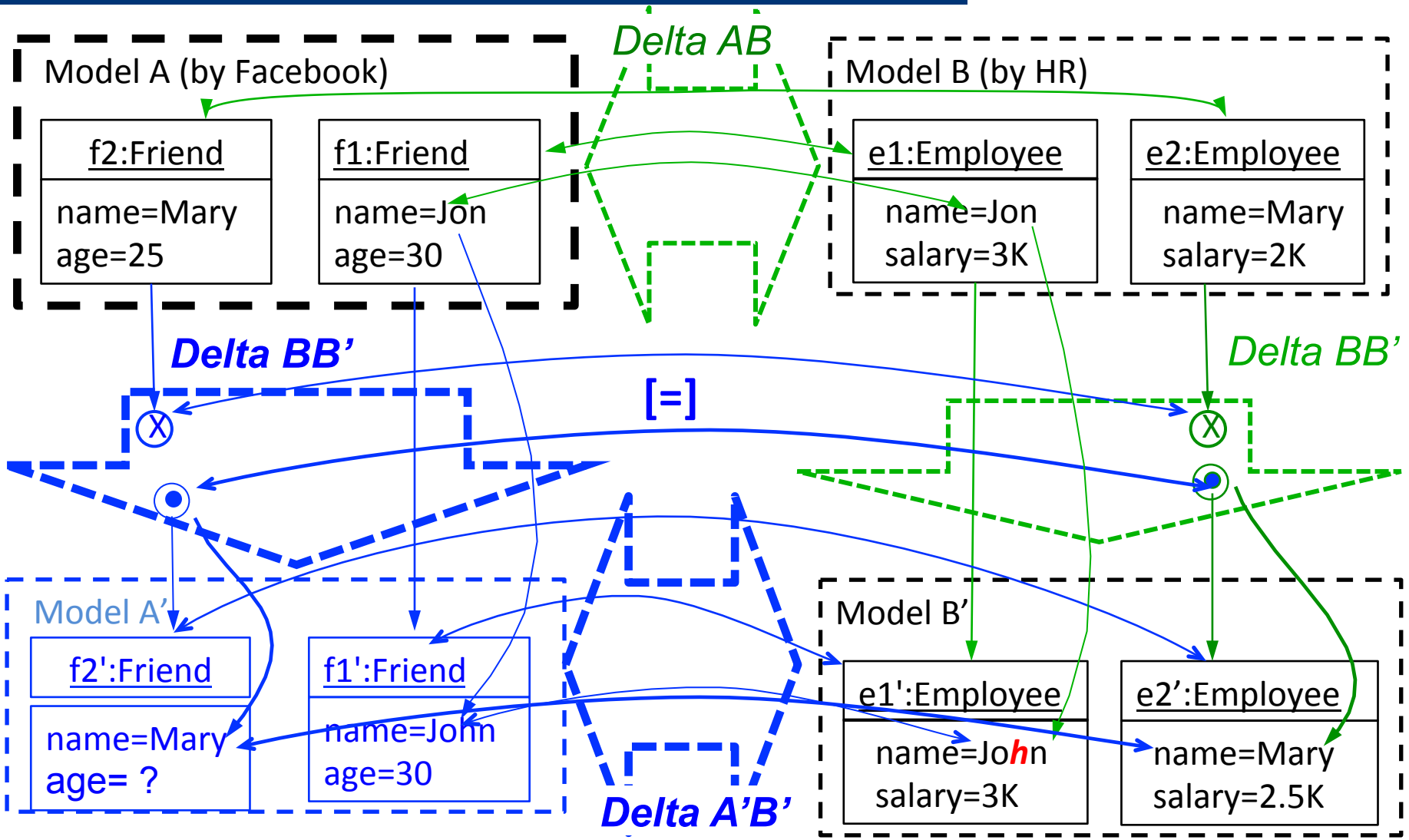
Attribute Propagation



Attribute Propagation



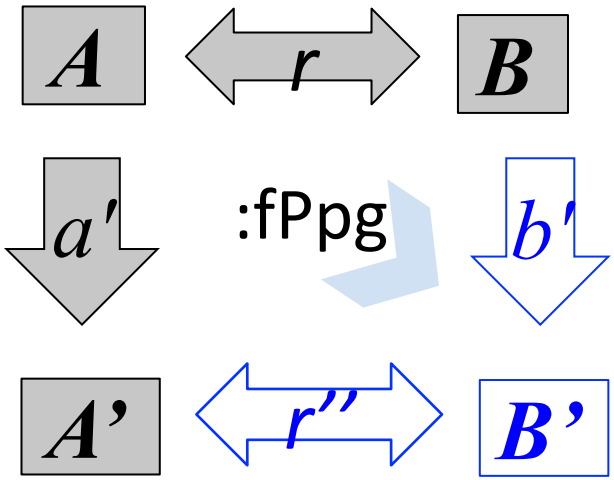
Delta Propagation



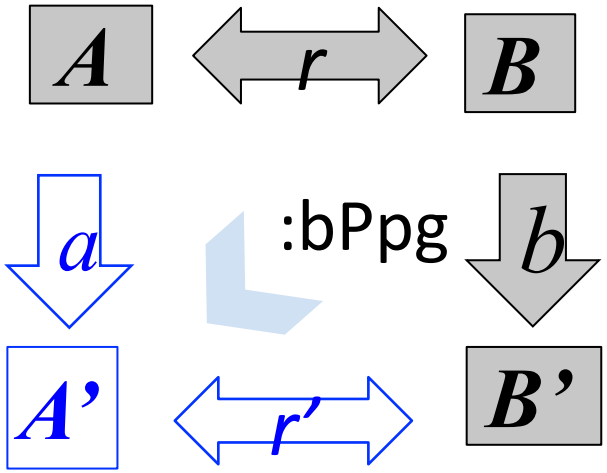
Lessons learned

- Delta discovery and delta propagation are separate concerns
  - The latter is much easier (algebra)
- Circularity/commutativity is fundamental
- Models with uncertainty are inevitable

Delta propagation abstractly

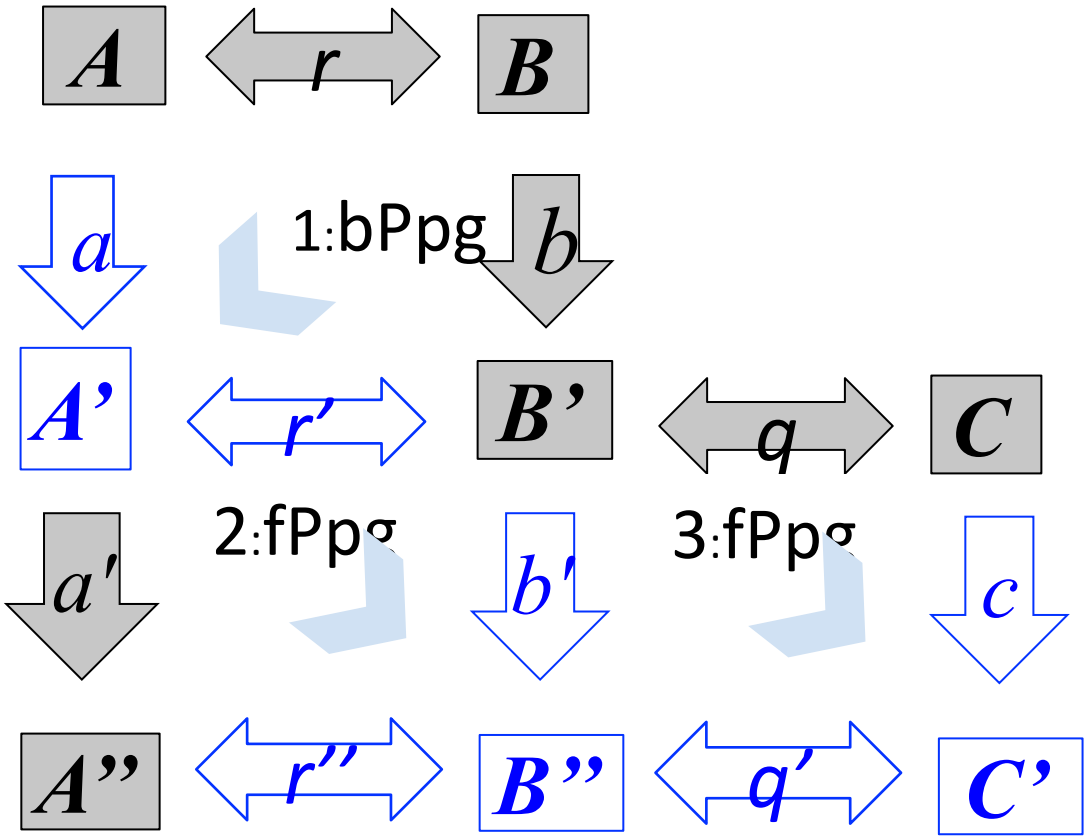


forward Propagation



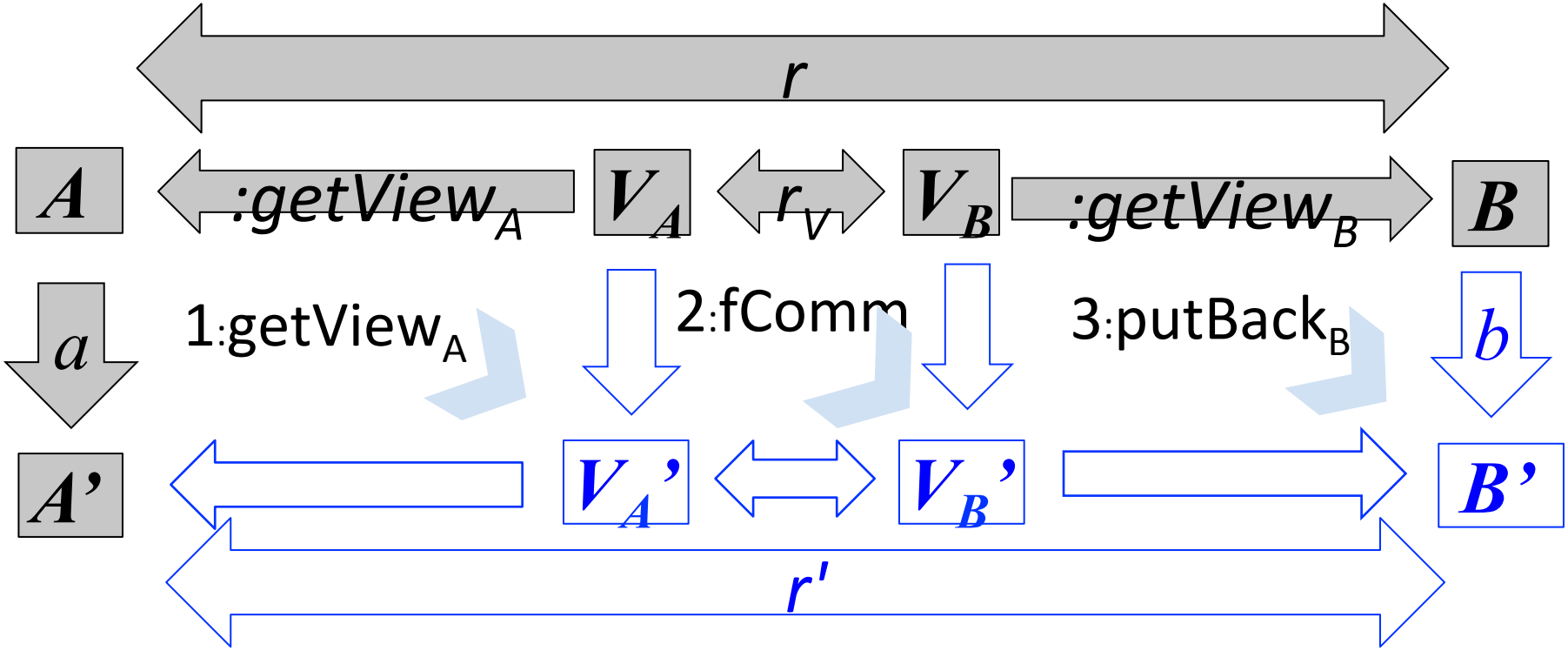
backward Propagation

Delta propagation in-the-large

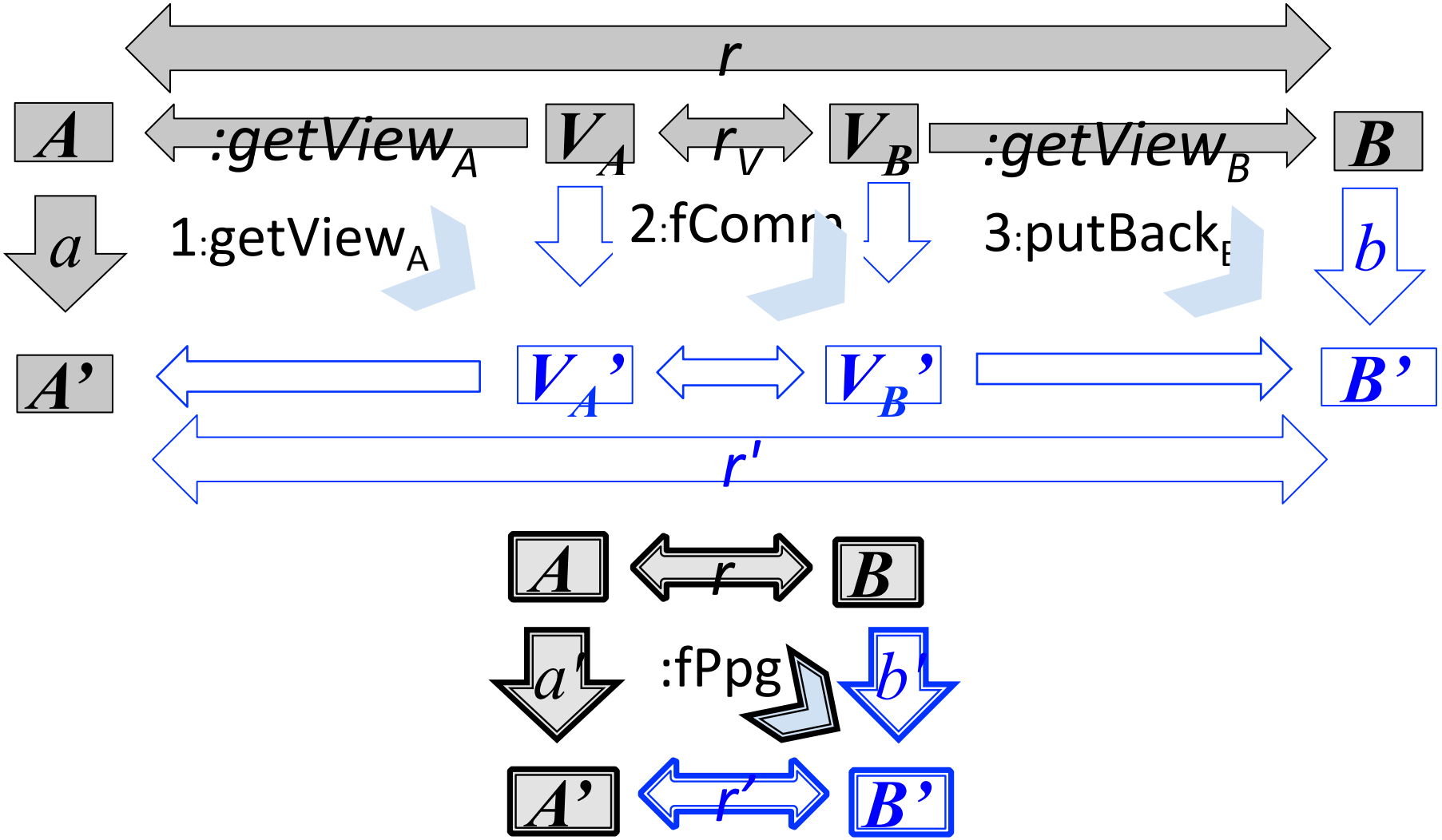


... and so on.

Delta propagation over views



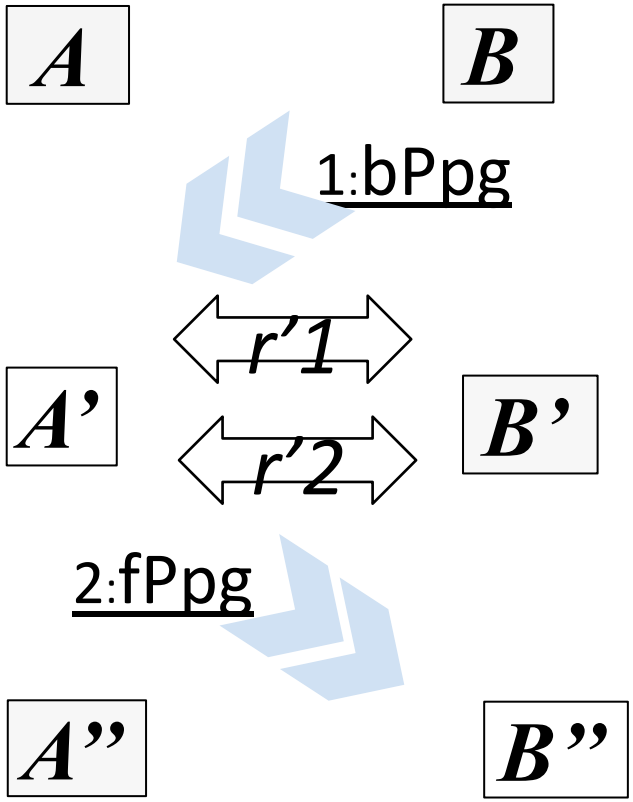
Delta propagation abstractly



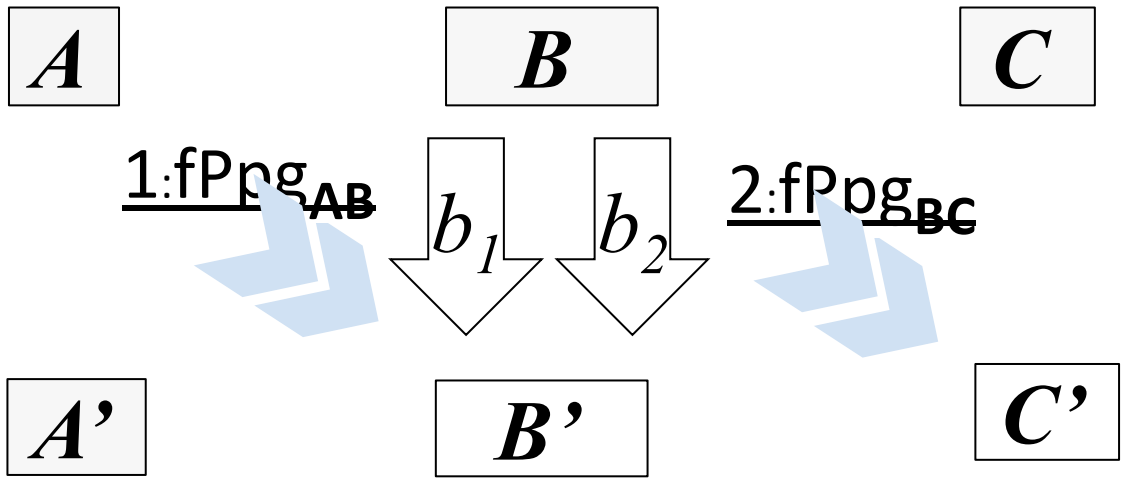
Lessons learned

- Delta discovery and delta propagation are separate concerns
  - The latter is much easier (algebra)
- Circularity/commutativity is fundamental
- Models with uncertainty are inevitable
- Deltas are important for proper composition
  - Tiling

State-based BX: erroneous vertical composition of Ppgs



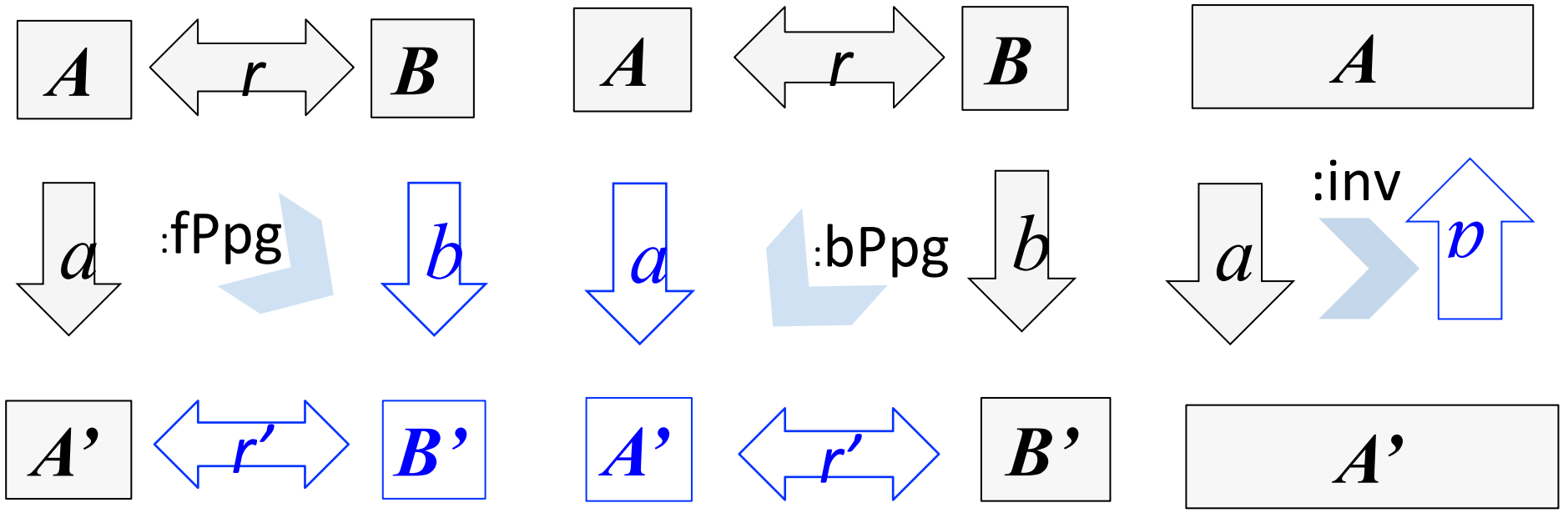
State-based BX: erroneous horizontal composition of Ppgs



Delta lenses

- Algebraic structures comprising
  - Basic operations over models and model deltas: delta composition, delta reversal, delta propagation, tile composition
  - Basic laws these operations and their composition should satisfy
- Product line: sync scenario ---> delta lens
- …are in active research
  - category theory

Delta Lenses: Operations

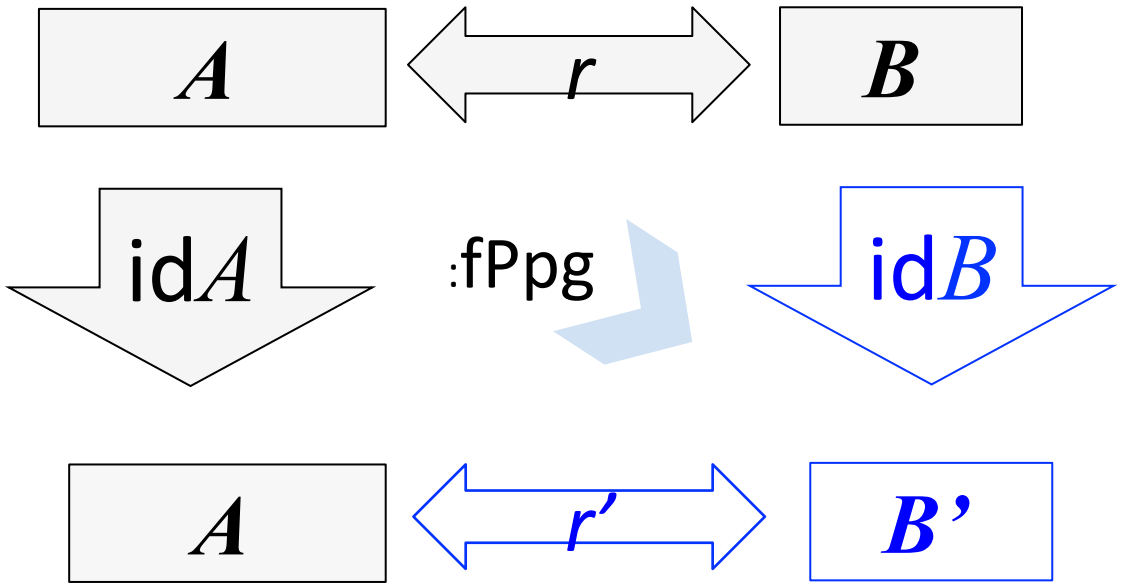


Forward update propagation

Backward update propagation

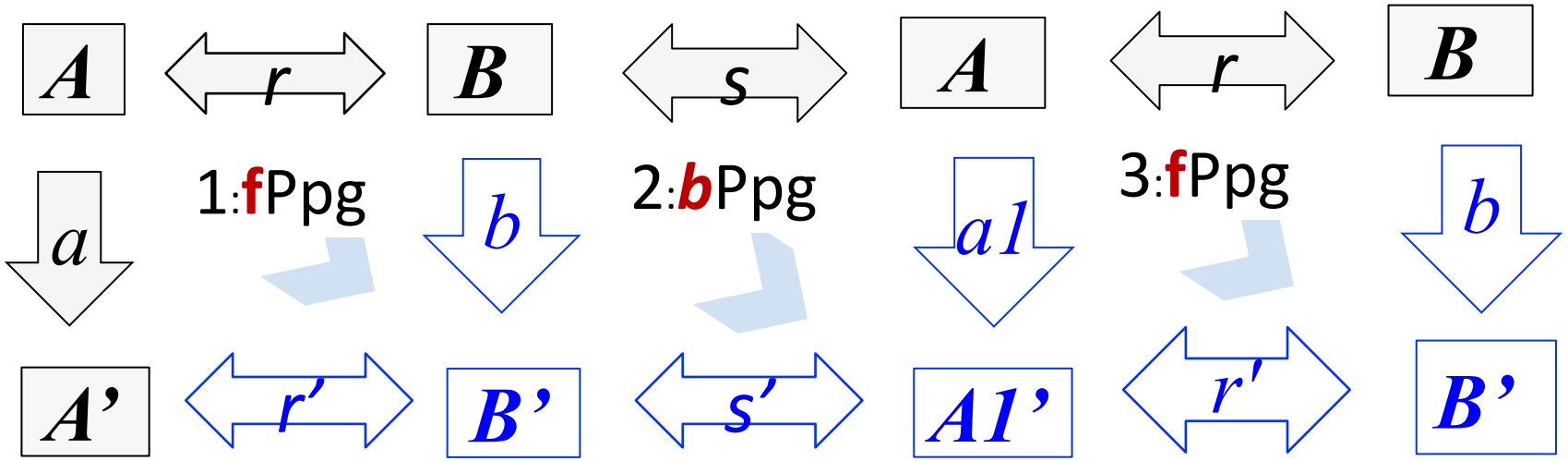
Update inversion

Delta Ppg laws: Identity propagation



Doing nothing is propagated
to doing nothing

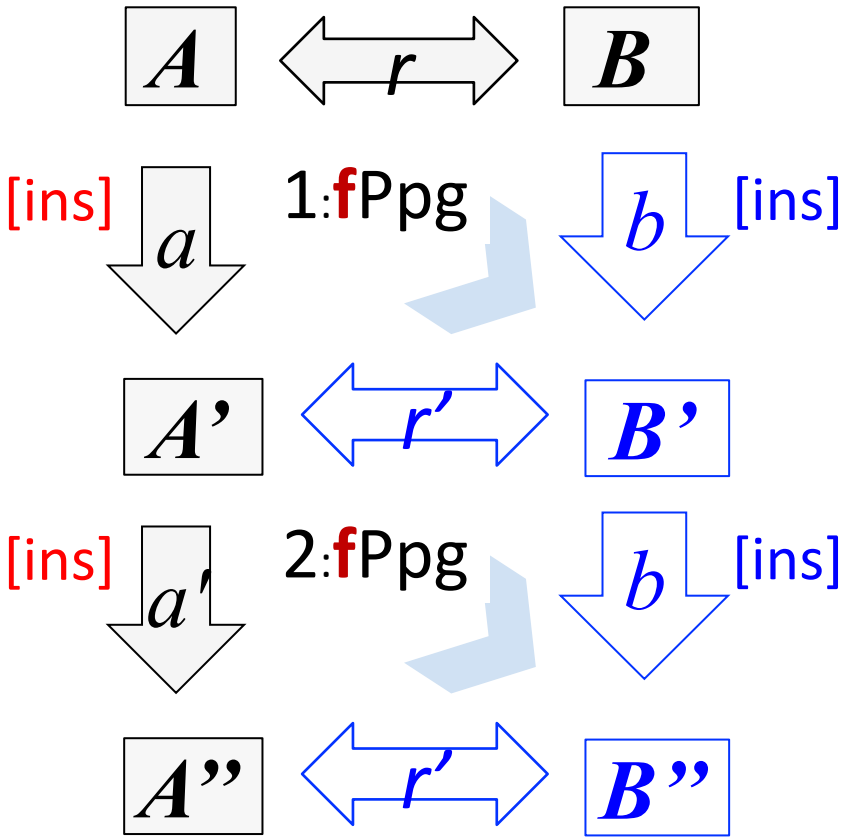
Delta Ppg laws: Invertibility (or round-tripping)

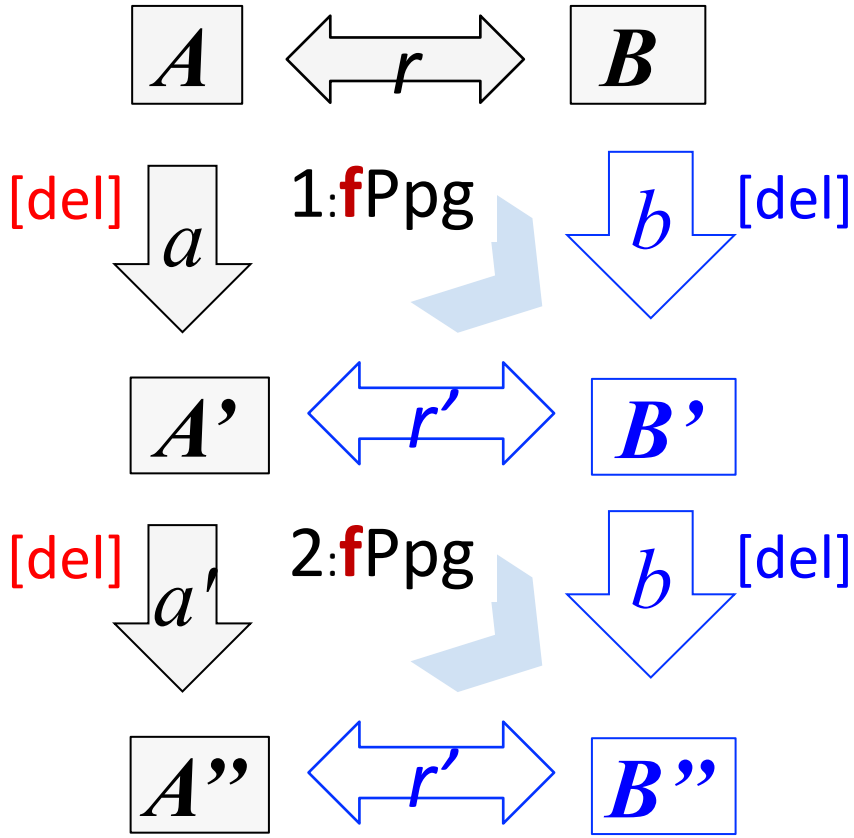


Weak invertibility: $a1 \neq a$

but $a1 \approx a$ in the following sense:

Delta Ppg laws: Monotonic PutPut





Delta lenses

- Algebraic structures comprising
  - Basic operations over models and model deltas: delta composition, delta reversal, delta propagation, tile composition
  - Basic laws these operations and their composition should satisfy
- Product line: sync scenario ---> delta lens
- …are in active research
  - category theory

Contents

- Foundations:
  - Terminology: model, metamodel, types of transformations, traceability.
  - Typical applications
- Conceptual and theoretical overview
- SE tools for BX
- Integration questions

Tools

- Medini QVT (QVT-like)
  - Currently unsupported, claims to be an implementation of QVT-R.
  - Diverges from the semantics of the QVT-R
    - (e.g., deletion of elements, no checkonly mode)
  - Open source (EPL)

Tools

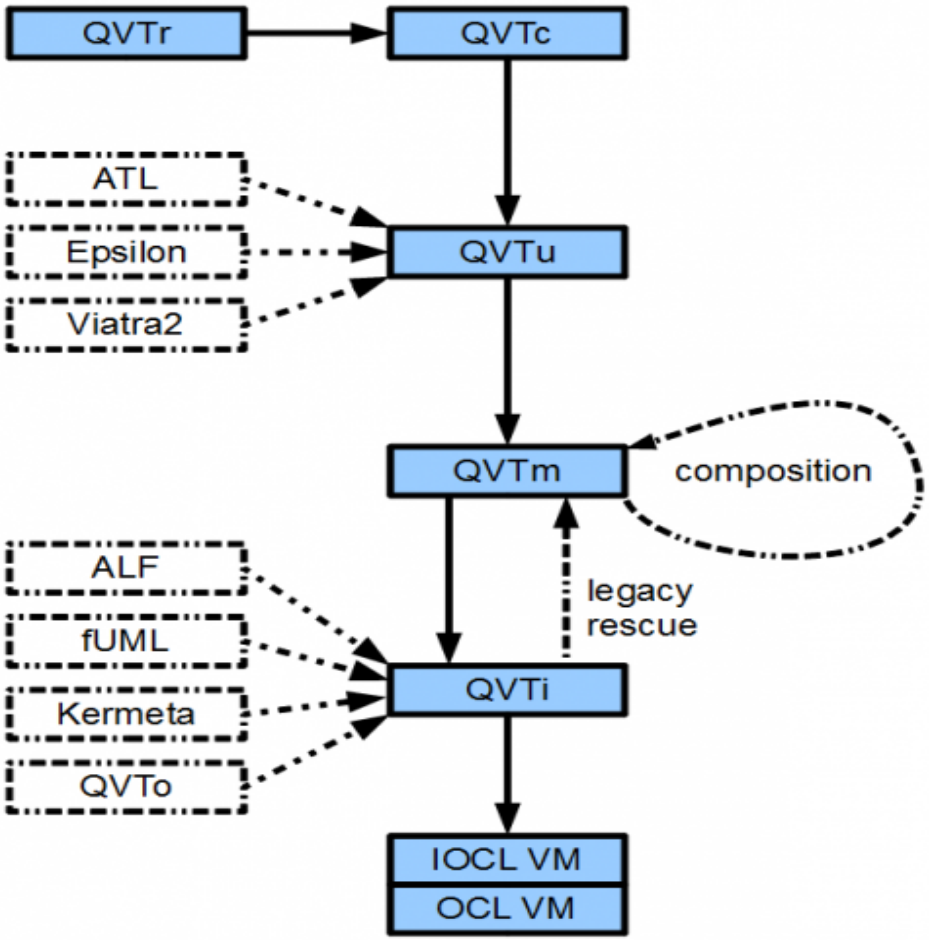
- ModelMorf:
  - From TCS, implementation of QVT-R circa 2007.
  - Pre-Beta is available to researchers; a proprietary version is available to commercial users (internal to TCS only).
  - Perdita's research shows it closely matches the QVT standard (and the intent of the standard).

Tools

- jQVT (QVT-like)
  - A QVT engine that is defined on the Java type system instead of EMF (uses Xbase language instead of OCL)
  - Generates native Java code from jQVT scripts.
  - Bidirectional???
- Echo:
  - Uses QVT-R for inter-model consistency and model repair.
  - Alloy model finder used to generate models.
  - Bidirectional.
  - Demo this week!

Tools

- Eclipse QVT
  - QVT-O is well defined and active (Eclipse M2M)
  - QVT-R and QVT-Core – work in progress (Ed Willink, York)



Tools

- XRound
  - Template-based bidirectional (asymmetric) language for XML-based models.
- XSugar
  - BX between two syntaxes for the same language; bijective.
- Epsilon (EWL+EVL) event-driven approach
  - Two update-in-place transformations (on source, target) executed when inter-model consistency rules are violated.

Integration

- *What can other communities learn from the SE approach to BX?*
  - Clear theory and pragmatic tools can co-exist
    - though tools that evidently implement the theory can be difficult to build
    - all manage trace-links in a systematic way [delta lenses]
  - The importance of metamodelling for understanding BX (typing is important).
    - semantics of transformations can be captured using metamodels.

Integration

- *What can you learn from other BX communities?*
  - How theory from other communities can be applied in understanding, improving and simplifying standard languages for BX in SE
    - QVT is still too complex (and proposed Eclipse refactoring of languages might make it even more complex!)
  - Key scenarios for bidirectionality (e.g., from DB community) – when do we really need it?

Integration

- *What do we do differently (or define differently)?*
  - We use operational transformations a lot
    - and model-to-concrete-syntax transformations
    - and model-to-text transformations
  - How do these fit into the BX space?
  - We use metamodels but PL community seems to do fine with grammars.
  - We have a hugely diverse set of modeling languages that need to be supported.

Integration

- *What concepts would we like to contribute to a unified theoretical framework for BX?*
  - Delta lens theory
  - Standard languages (QVT-R, OCL, Ecore) that both exist and are widely used.
  - Support for traceability (trace-links as a side effect of running model management operations), both as a pragmatic tool and as a theoretical basis.