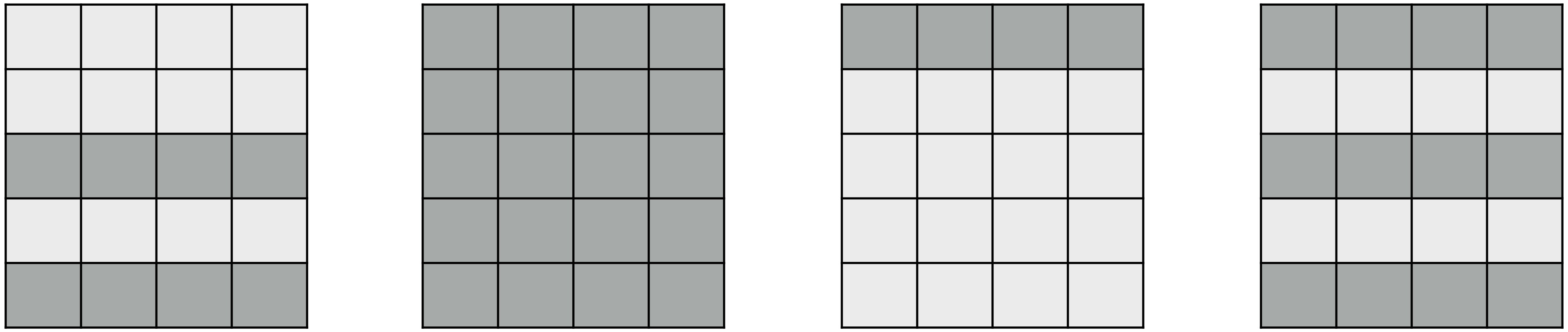
# Update Policies

Abhijeet Mohapatra, Sudhir Agarwal, Michael Genesereth

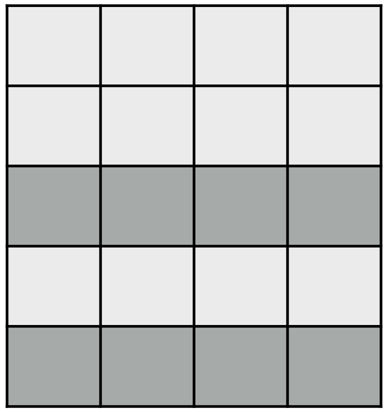
{abhijeet, sudhir, genesereth} @ cs.stanford.edu

# Databases

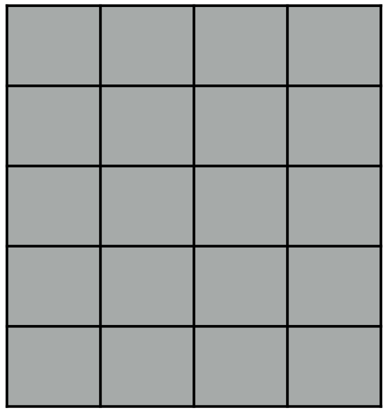


# Databases

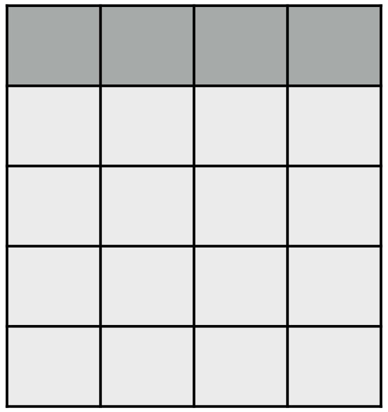
***Constraints*** characterize ***legal states***



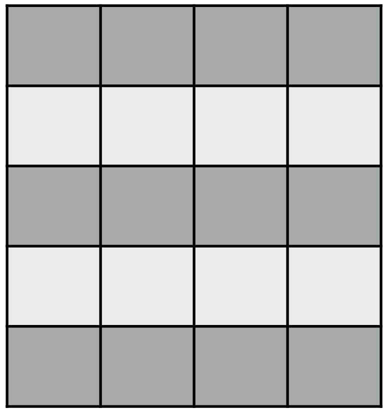
✓ Legal



❌ Illegal



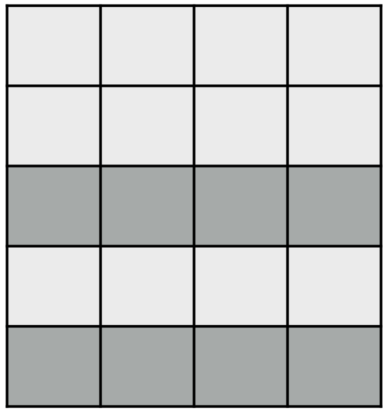
✓ Legal



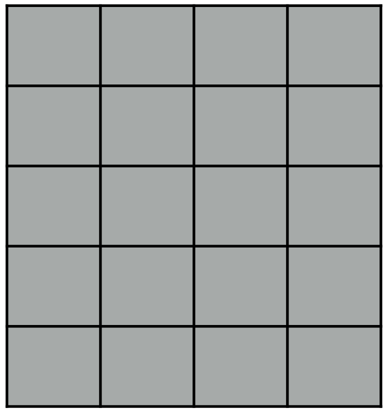
✓ Legal

# Databases

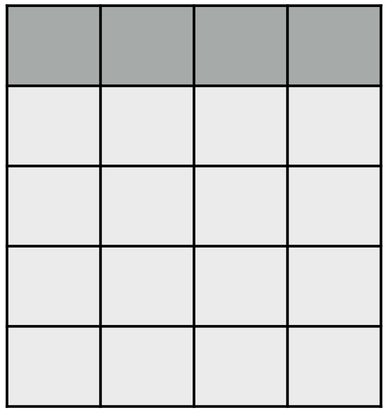
***Constraints*** characterize ***legal states***



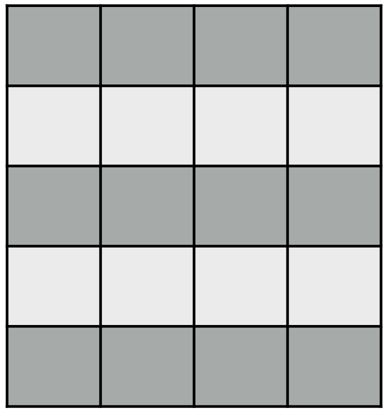
✓ Legal



❌ Illegal

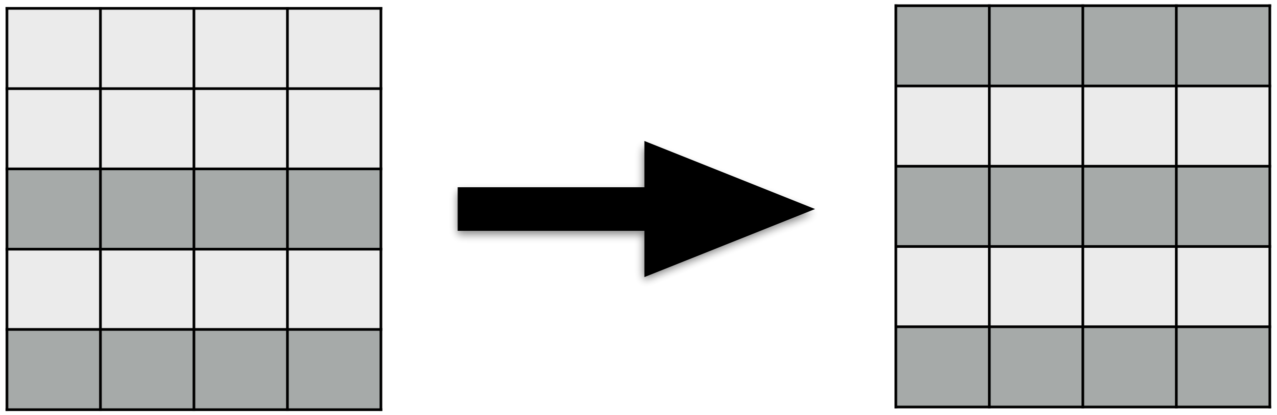


✓ Legal



✓ Legal

***Transactions*** update database states



# Underspecified Transactions

To simplify the data entry process, users may under specify a transaction.

***Take at most 2 courses*** *from*
*CS 157, PHIL 161, or STATS 116*

# Underspecified Transactions

To simplify the data entry process, users may under specify a transaction.

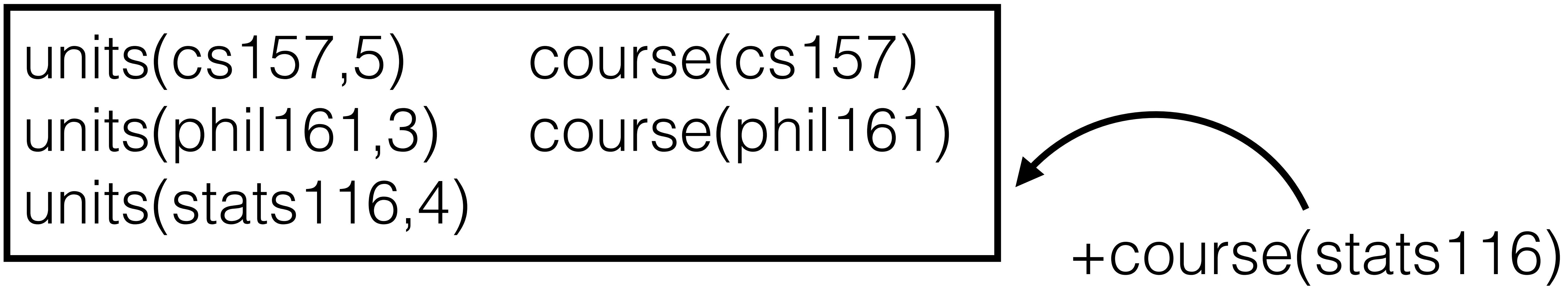
***Take at most 2 courses** from*
*CS 157, PHIL 161, or STATS 116*

```
units(cs157,5)       course(cs157)
units(phil161,3)     course(phil161)
units(stats116,4)
```

# Underspecified Transactions

To simplify the data entry process, users may under specify a transaction.

***Take at most 2 courses*** *from*
*CS 157, PHIL 161, or STATS 116*



units(cs157,5) course(cs157)
units(phil161,3) course(phil161)
units(stats116,4)

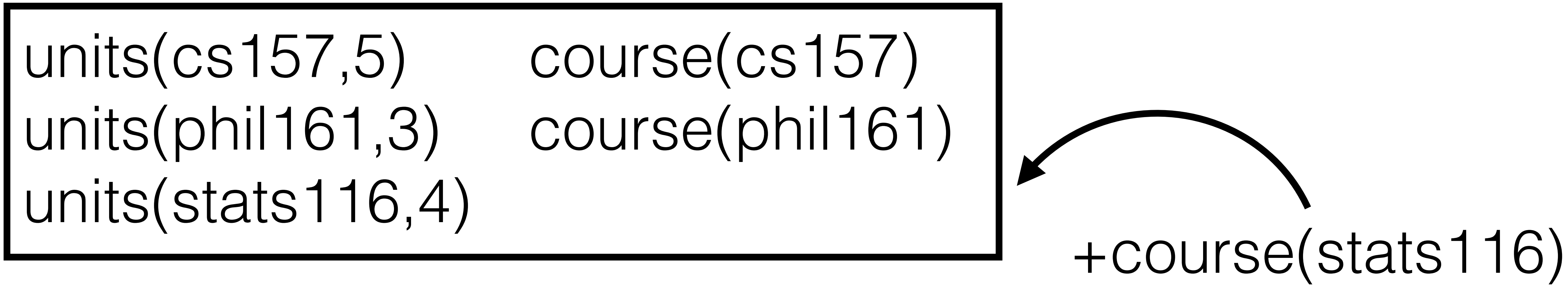
+course(stats116)

*What happens when a user adds STATS 116 as a selection?*

# Underspecified Transactions

To simplify the data entry process, users may under specify a transaction.

***Take at most 2 courses*** *from*
*CS 157, PHIL 161, or STATS 116*



units(cs157,5)
units(phil161,3)
units(stats116,4)
course(cs157)
course(phil161)

+course(stats116)

*What happens when a user adds STATS 116 as  a selection?*

Remove PHIL 161? Remove CS 157? Remove Both? Reject STATS 116?

*Constraints are not enough!*

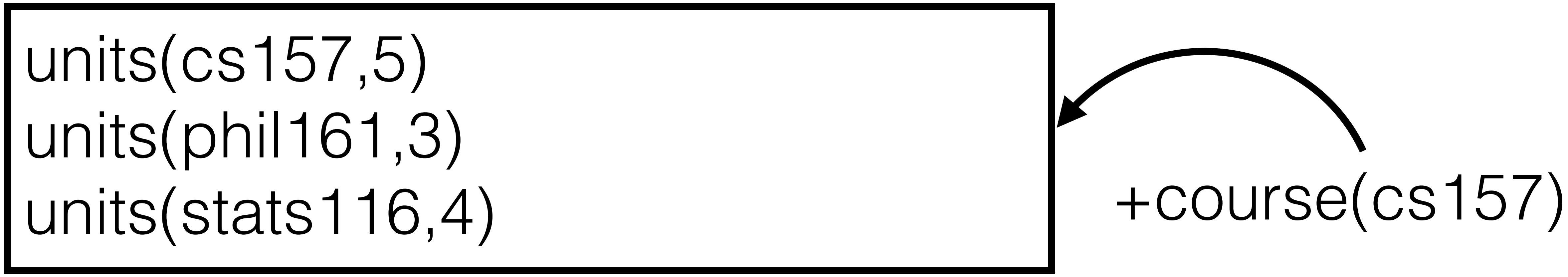
# Administrator's Preferences

A database administrator may want to explicitly specify how to complete the underspecified transaction.

```
units(cs157,5)
units(phil161,3)
units(stats116,4)
```

# Administrator's Preferences

A database administrator may want to explicitly specify how to complete the underspecified transaction.



units(cs157,5)
units(phil161,3)
units(stats116,4)

+course(cs157)

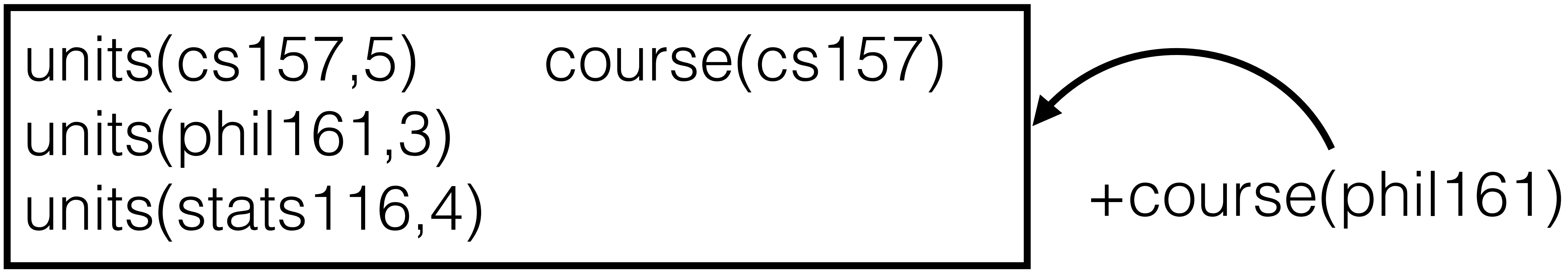
# Administrator's Preferences

A database administrator may want to explicitly specify how to complete the underspecified transaction.

```
units(cs157,5)        course(cs157)
units(phil161,3)
units(stats116,4)
```

# Administrator’s Preferences

A database administrator may want to explicitly specify how to complete the underspecified transaction.



units(cs157,5)
units(phil161,3)
units(stats116,4)
course(cs157)

+course(phil161)

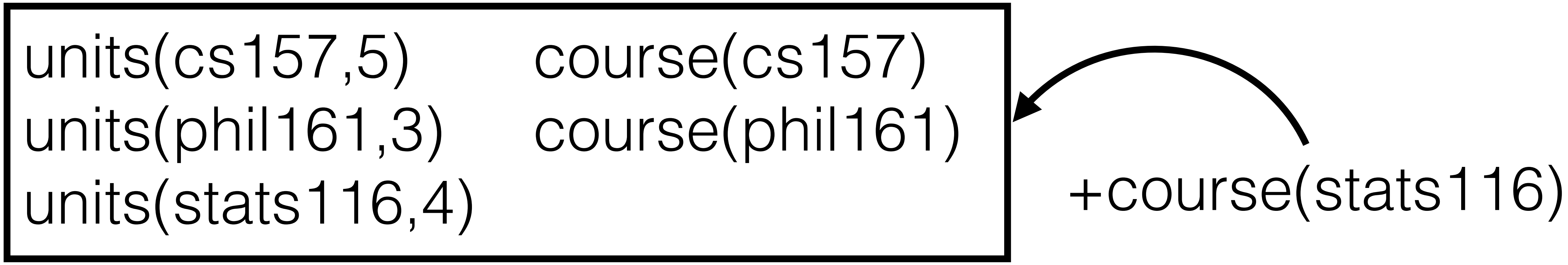
# Administrator's Preferences

A database administrator may want to explicitly specify how to complete the underspecified transaction.

```
units(cs157,5)        course(cs157)
units(phil161,3)      course(phil161)
units(stats116,4)
```

# Administrator’s Preferences

A database administrator may want to explicitly specify how to complete the underspecified transaction.

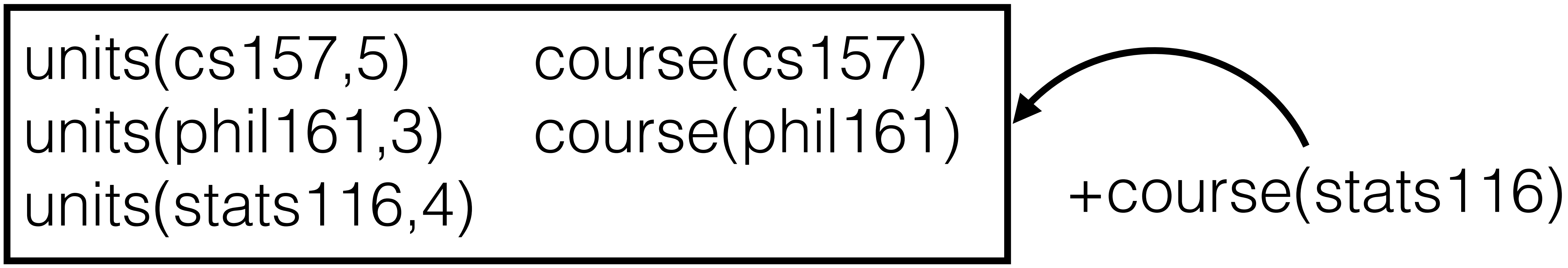


units(cs157,5) course(cs157)
units(phil161,3) course(phil161)
units(stats116,4)

+course(stats116)

# Administrator’s Preferences

A database administrator may want to explicitly specify how to complete the underspecified transaction.



units(cs157,5)
units(phil161,3)
units(stats116,4)

course(cs157)
course(phil161)

+course(stats116)

*Retain the most recent course selection*

# Administrator's Preferences

A database administrator may want to explicitly specify how to complete the underspecified transaction.

units(cs157,5)
units(phil161,3)
units(stats116,4)
course(stats116)
course(phil161)

*Retain the most recent course selection*

# Administrator's Preferences

A database administrator may want to explicitly specify how to complete the underspecified transaction.

```
units(cs157,5)        course(stats116)
units(phil161,3)      course(phil161)
units(stats116,4)
```

*Retain the most recent course selection*

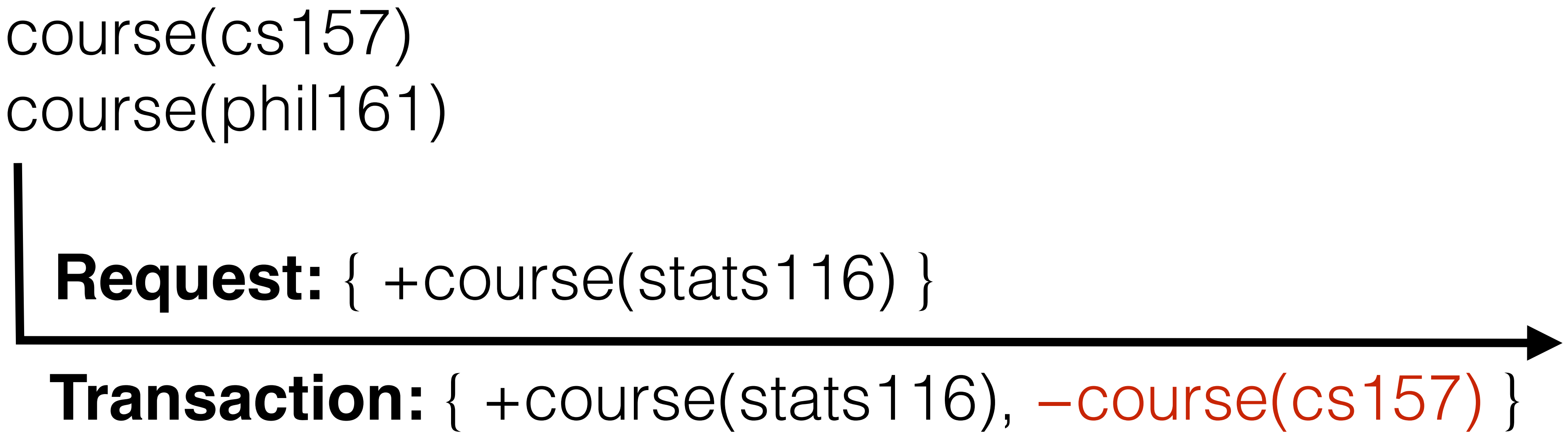
What is needed is the ability to characterize ***what the user wants to be changed***, and ***what should be changed*** across states.

# Update Policies

course(cs157)
course(phil161)

**Request:** { +course(stats116) }

# Update Policies

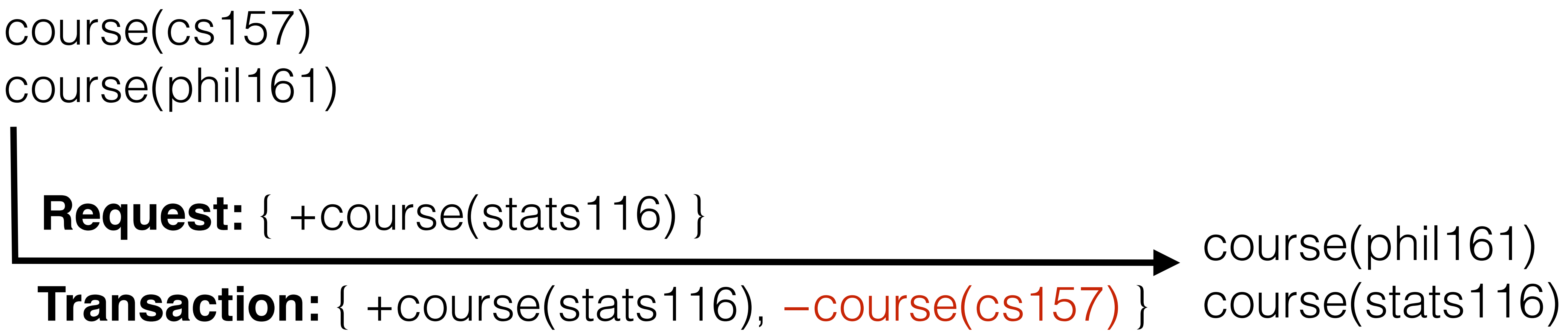


course(cs157)
course(phil161)

**Request:** { +course(stats116) }

**Transaction:** { +course(stats116), –course(cs157) }

# Update Policies



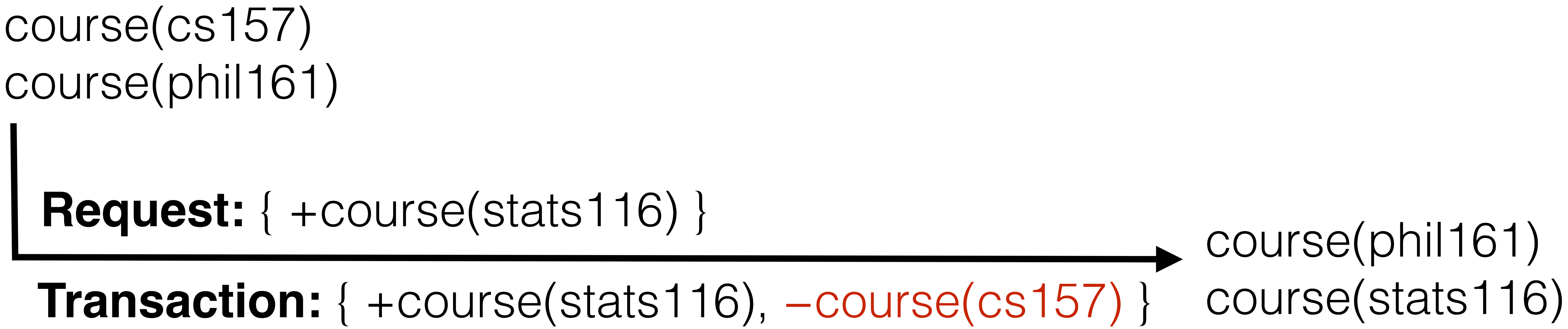
course(cs157)
course(phil161)

**Request:** { +course(stats116) }

**Transaction:** { +course(stats116), –course(cs157) }

course(phil161)
course(stats116)

# Update Policies



course(cs157)
course(phil161)

**Request:** { +course(stats116) }

**Transaction:** { +course(stats116), –course(cs157) }

course(phil161)
course(stats116)

A ***legal transaction*** is a mapping between two legal states of a database

An ***update policy*** is a mapping of the form

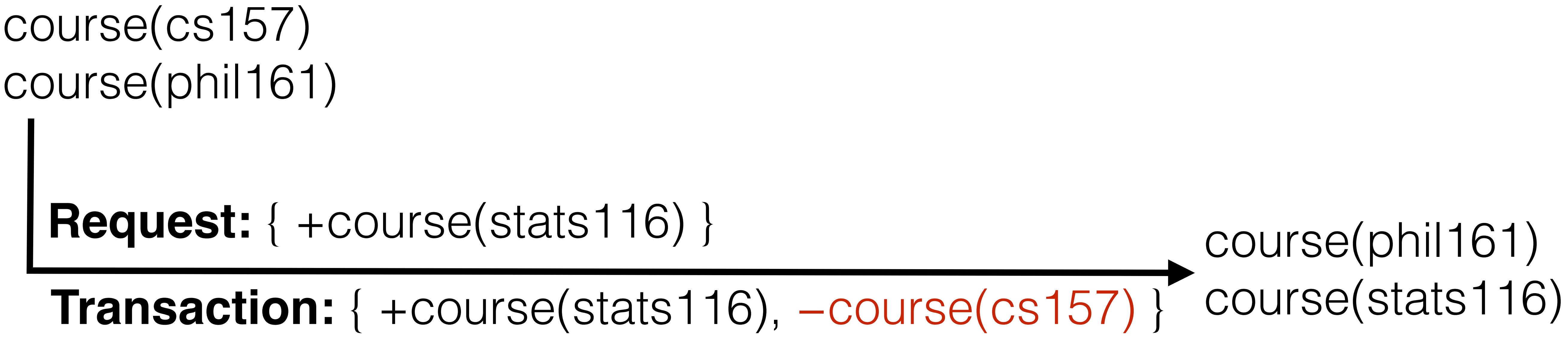
Legal State **x** Request → Legal Transaction

# Inclusive Policies

A policy is inclusive if for every request on a legal state, the completed transaction is a superset of the request.

# Inclusive Policies

A policy is inclusive if for every request on a legal state, the completed transaction is a superset of the request.



course(cs157)
course(phil161)

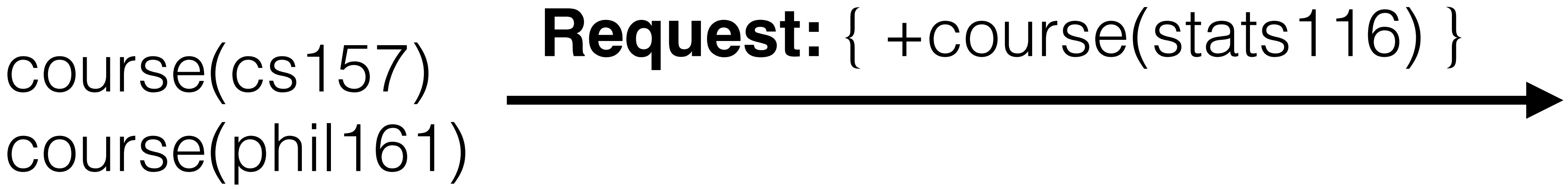
**Request:** { +course(stats116) }

**Transaction:** { +course(stats116), −course(cs157) }

course(phil161)
course(stats116)

*Inclusive policy*

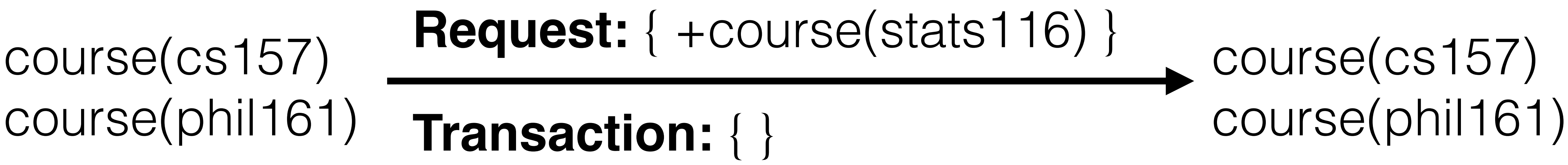
# Non-Inclusive Policies



course(cs157)
course(phil161)

**Request:** { +course(stats116) }

# Non-Inclusive Policies



course(cs157)
course(phil161)

**Request:** { +course(stats116) }

**Transaction:** { }

course(cs157)
course(phil161)

*Transaction rollback*

# Minimal Inclusive Policies

Let P be an Inclusive Policy.

For every *legal state* S, *transaction request* R, let $f_P(S, R)$ denote the completed transaction.

P is minimal iff no non-empty subset of $f_P(S, R)$ is a legal transaction.

Correspond to *necessary and sufficient repairs*

# Minimal Inclusive Policies

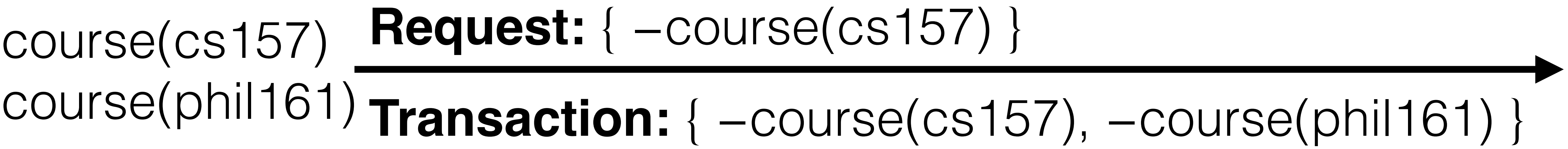
Let P be an Inclusive Policy.

For every *legal state* S, *transaction request* R, let $f_P(S, R)$ denote the completed transaction.

P is minimal iff no non-empty subset of $f_P(S, R)$ is a legal transaction.

Correspond to *necessary and sufficient repairs*

***Non-minimal Inclusive Policy***
e.g. *add all or delete all*



course(cs157)
course(phil161)

**Request:** { −course(cs157) }

**Transaction:** { −course(cs157), −course(phil161) }

# Dynamic Datalog

- Datalog is a function-free subset of PROLOG.
  ancestor(X, Y) :- parent(X, Y)
  ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y)

- Dynamic Datalog extends standard Datalog with negation as failure (¬) to support update policies.

- Four update operators are introduced: $\delta^-$, $\delta^+$, $\Delta^-$, and $\Delta^+$.

  $\boldsymbol{\delta^-} \boldsymbol{\phi}$ and $\boldsymbol{\delta^+} \boldsymbol{\phi}$ characterize **transaction requests** to delete and add $\boldsymbol{\phi}$ respectively.

  $\boldsymbol{\Delta^-} \boldsymbol{\phi}$ and $\boldsymbol{\Delta^+} \boldsymbol{\phi}$ characterize deletion and insertion of $\boldsymbol{\phi}$ in a **transaction** respectively.

# Example 1

*Remove all selected courses*

**Update as per user's request. (Inclusive Policy)**

**If two courses are currently selected, and a new selection is made, then remove all selections.**

# Example 1

*Remove all selected courses*

**Update as per user's request. (Inclusive Policy)**

$\Delta^{+}$ course(X) :- $\delta^{+}$course(X)

$\Delta^{-}$ course(X) :- $\delta^{-}$course(X)

**If two courses are currently selected, and a new selection is made, then remove all selections.**

$\Delta^{-}$ course(Y) :- course(Y), course(Z), Y $\neq$ Z, $\delta^{+}$course(X)

$\Delta^{-}$ course(Z) :- course(Y), course(Z), Y $\neq$ Z, $\delta^{+}$course(X)

# Example 2

*Retain the course with greatest number of units*

**Update as per user's request. (Inclusive Policy)**

**If two courses are currently selected, and a new selection is made, then remove the selection with fewer units.**

# Example 2

*Retain the course with greatest number of units*

**Update as per user's request. (Inclusive Policy)**

$\Delta^{+}$ course(X) :- $\delta^{+}$course(X)

$\Delta^{-}$ course(X) :- $\delta^{-}$course(X)

**If two courses are currently selected, and a new selection is made, then remove the selection with fewer units.**

$\Delta^{-}$ course(Y) :- course(Y), course(Z), $\delta^{+}$course(X),
units(Y, U), units(Z, V), $U < V$

# Example 3

*Retain the most recent course selection*

**Update as per user's request (Inclusive Policy)**

**Maintain the most recently added course**

**If two courses are currently selected, remove all but the most recent addition**

# Example 3

*Retain the most recent course selection*

**Update as per user's request (Inclusive Policy)**

$\Delta^{+}\,\mathrm{course}(X) \;\text{:-}\; \delta^{+}\mathrm{course}(X)$

$\Delta^{-}\,\mathrm{course}(X) \;\text{:-}\; \delta^{-}\mathrm{course}(X)$

**Maintain the most recently added course**

$\Delta^{+}\,\mathrm{recent}(X) \;\text{:-}\; \delta^{+}\mathrm{course}(X)$

$\Delta^{-}\,\mathrm{recent}(Y) \;\text{:-}\; \mathrm{recent}(Y),\ \delta^{+}\mathrm{course}(X)$

$\Delta^{-}\,\mathrm{recent}(X) \;\text{:-}\; \mathrm{recent}(X),\ \delta^{-}\mathrm{course}(X)$

**If two courses are currently selected, remove all but the most recent addition**

$$\begin{aligned}\Delta^{-}\,\mathrm{course}(Y) \;\text{:-}\; &\mathrm{course}(Y),\ \mathrm{course}(Z),\ Y \neq Z,\\ &\delta^{+}\mathrm{course}(X),\ \neg\mathrm{recent}(Y)\end{aligned}$$

# Discussion

- Related Work
  - Event-Condition-Action (ECA) Rules: non-deterministic, less expressive.
  - STRIPS: expressible in Dynamic Datalog.
  - Transaction Repair: administrator preferences not included.

- Dynamic Datalog (with arithmetic builtins) can express every computable update policy on ordered domains.

- Verification of policies:
  - Undecidable in general.
  - Decidable for bounded domains, or non-recursive rules.
  - Resolution-based procedure proposed in paper.

# Prior Work

- ECA rules cannot express the following update policies.

  Example 1. $\Delta^{+}q(X)$ :- $\delta^{+}p(X)$, $\delta^{-}r(X)$

  Example 2. $\Delta^{+}r$ :- $\delta^{+}p$, $\delta^{+}q$
  $\Delta^{+}q$ :- $\delta^{+}p$, $\neg\delta^{+}q$

# Prior Work

- ECA rules cannot express the following update policies.

  Example 1. $\Delta^{+}q(X)$ :- $\delta^{+}p(X)$, $\delta^{-}r(X)$

  Example 2. $\Delta^{+}r$ :- $\delta^{+}p$, $\delta^{+}q$
  $\Delta^{+}q$ :- $\delta^{+}p$, $\neg\delta^{+}q$

- STRIPS encoding in Dynamic Datalog.
  - Action a encoded as $\delta^{+}a$
  - $x \in$ Add list encoded as $\Delta^{+}x$
  - $x \in$ Delete list encoded as $\Delta^{-}x$