Locking for Concurrent Transactions on Ontologies

Matthias Thimm

joint work with Stefan Scheglmann, Steffen Staab, and Gerd Gröner

Definition

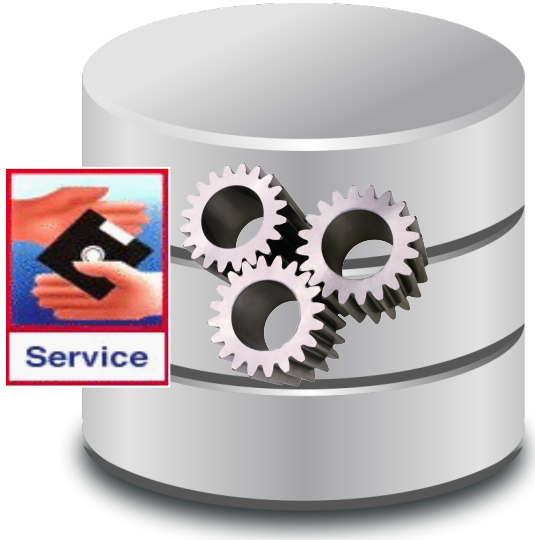
Definition:

Transaction
- begin of transaction
- sequence of operations
- end of transaction

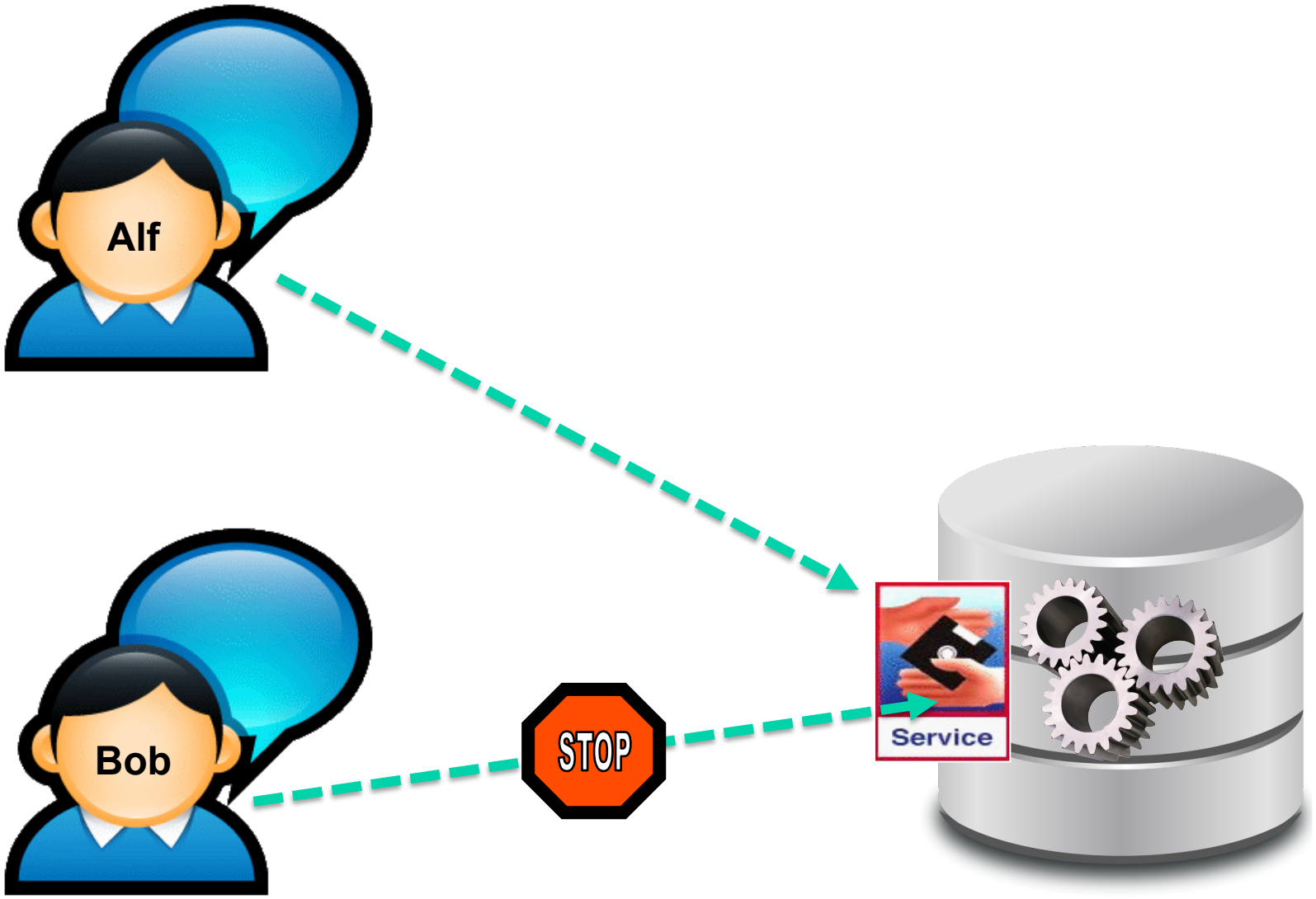
Critical Operations
- tell
- forget
- ask

critical operation

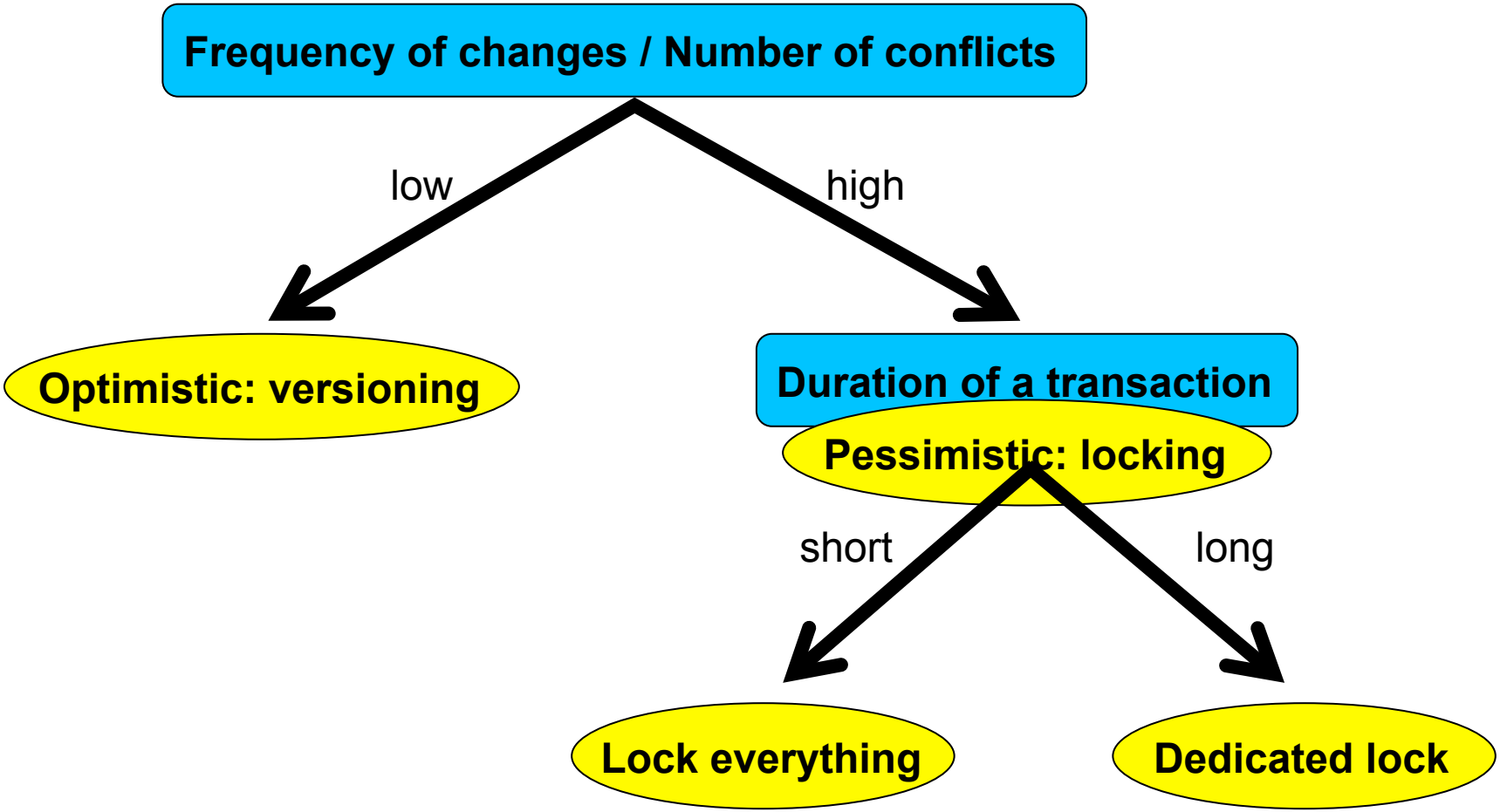
Non-critical operation
- user input
- response
- etc



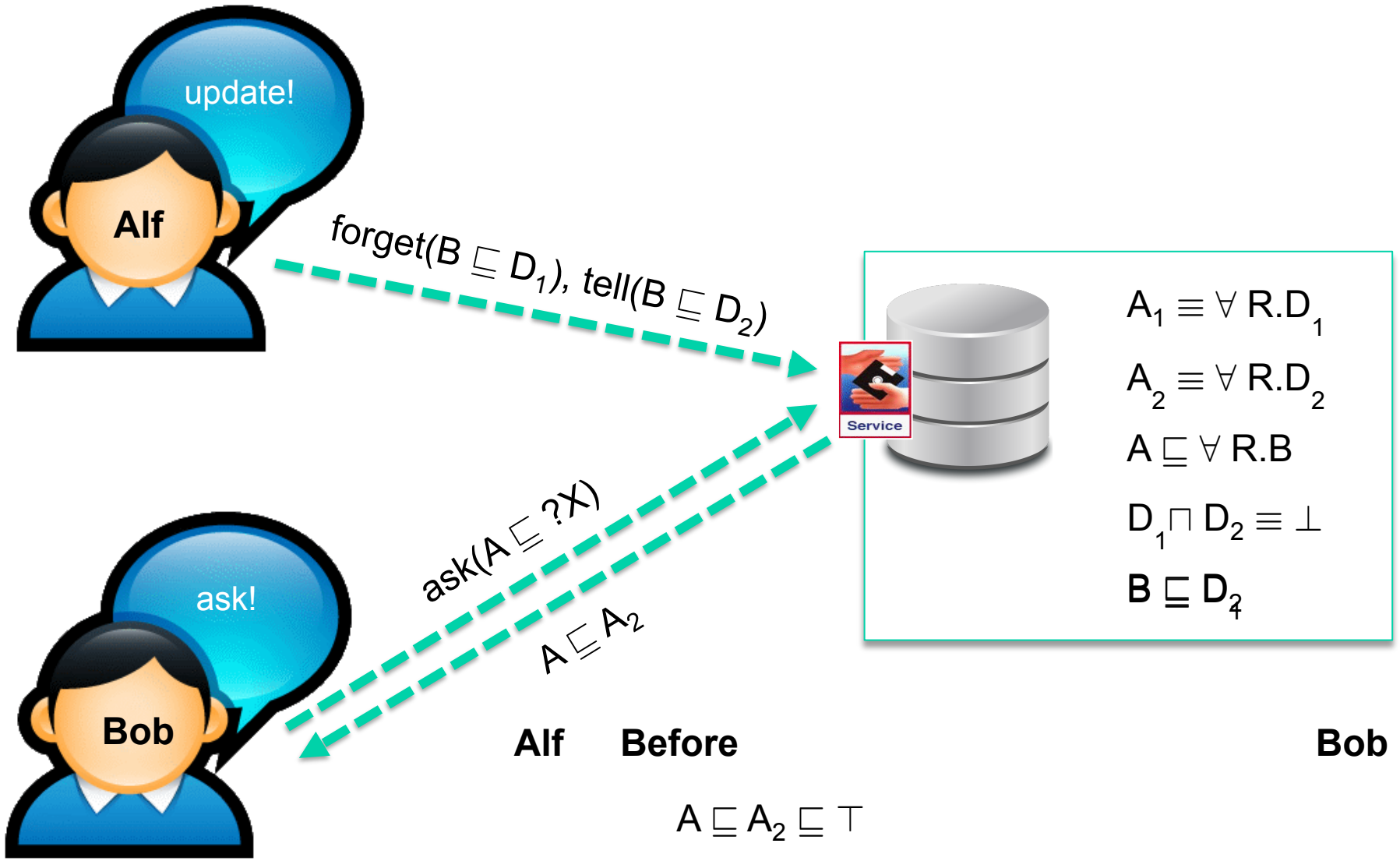
Setting



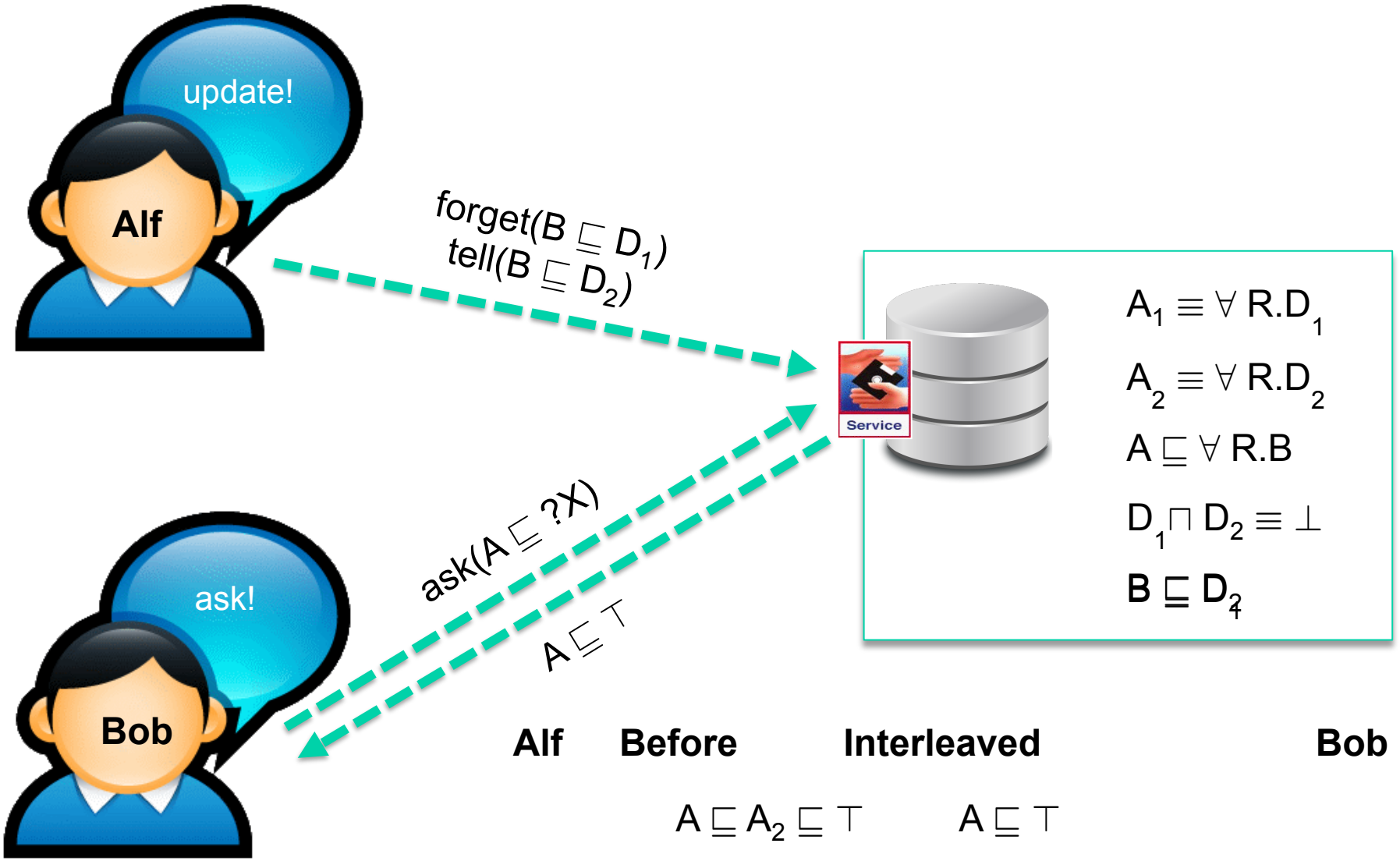
Problem Dimensions



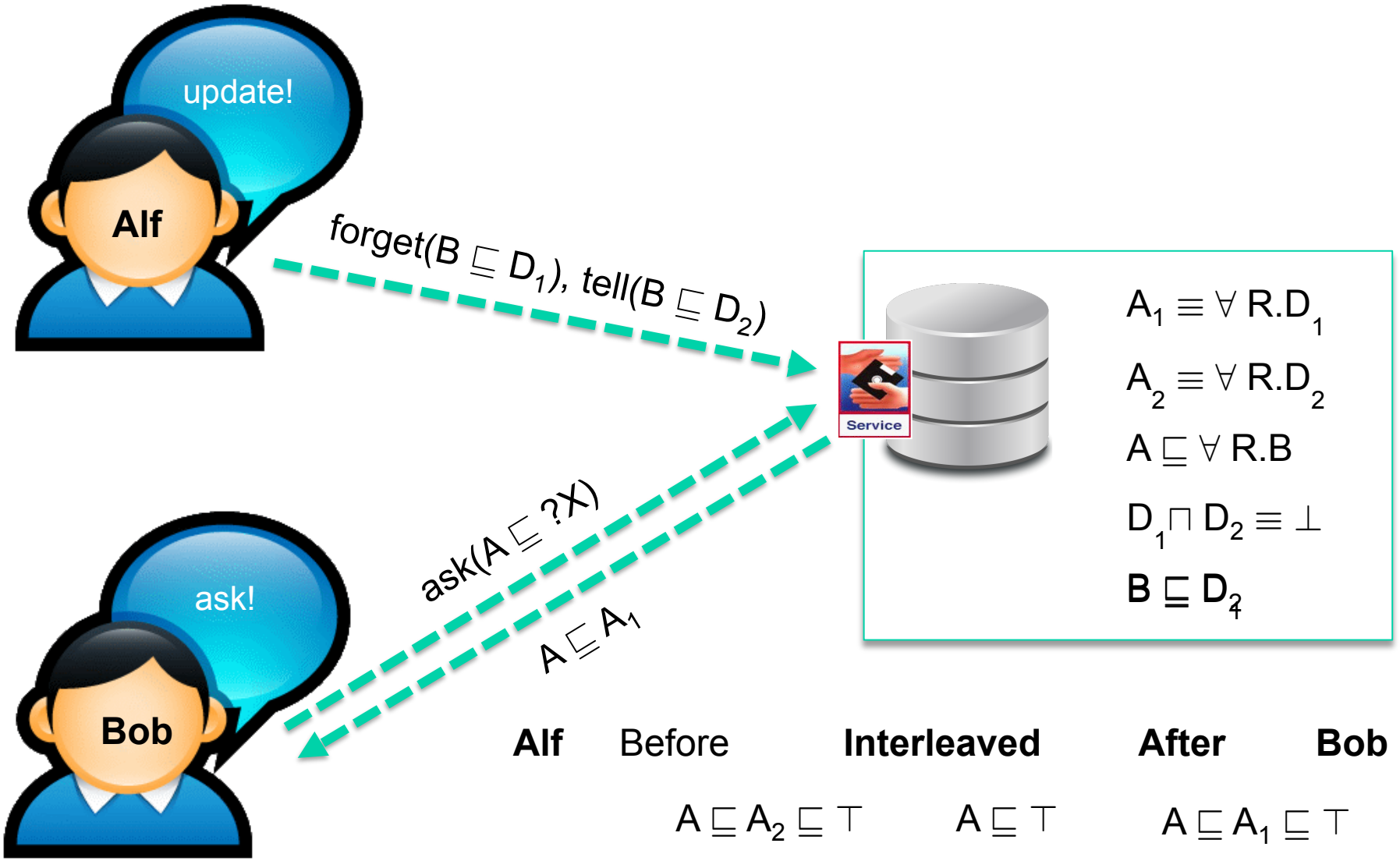
Problem


update!
Alf
forget(B ⊑ D1), tell(B ⊑ D2)
Service
A1 ≡ ∀ R.D1
A2 ≡ ∀ R.D2
A ⊑ ∀ R.B
D1 ⊓ D2 ≡ ⊥
B ⊑ D2
ask!
Bob
ask(A ⊑ ?X)
A ⊑ A2
Alf
Before
Bob
A ⊑ A2 ⊑ ⊤

Intermediate



After

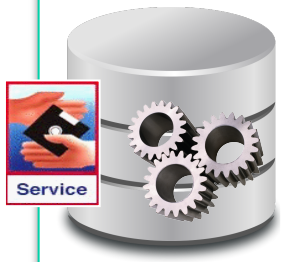

update!
Alf
forget(B ⊑ D1), tell(B ⊑ D2)
Service
A1 ≡ ∀ R.D1
A2 ≡ ∀ R.D2
A ⊑ ∀ R.B
D1 ⊓ D2 ≡ ⊥
B ⊑ D2
ask!
Bob
ask(A ⊑ ?X)
A ⊑ A1
Alf
Before
Interleaved
After
Bob
A ⊑ A2 ⊑ ⊤
A ⊑ ⊤
A ⊑ A1 ⊑ ⊤

LOCKING FOR ONTOLOGY-ACCESS

Transaction Processing I

$forget(B \sqsubseteq D_1)$

begin



$A_1 \equiv \forall R.D_1$

$A_2 \equiv \forall R.D_2$

$A \sqsubseteq \forall R.B$

$D_1 \sqcap D_2 \equiv \bot$

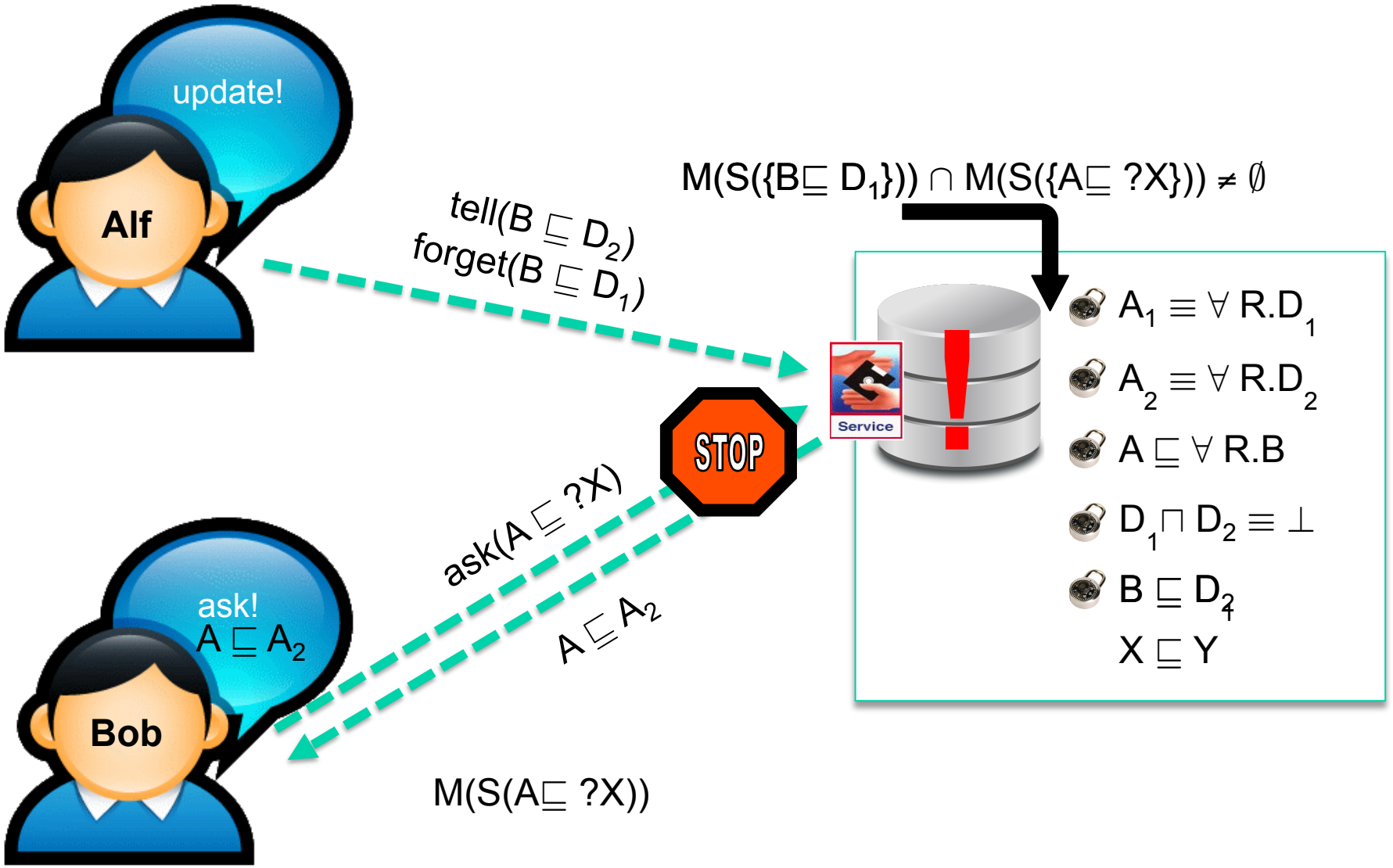
$B \sqsubseteq D_1$

$X \sqsubseteq Y$

Core idea: Ontology module based locks

- Syntactical locality based module approximation
- Function S()
  - given a set of axioms
  - provides us with a signature
- Function M()
  - given a signature
  - provides us with a module for S

Transaction Processing II



Evaluation: Construction of Histories

Input:

- National Cancer Thesaurus (NCIt)
  - OWL EL++ Ontology
  - 4 consecutive versions
  - ~600K Axioms,
    - 500K annotation, 38K classes, 90 object properties

Steps:

1. Identifying 420 Transactions of 6-12 operations
   - How: From syntactic diffs of consecutive versions
2. Each critical operation followed by non-critical with same duration
3. Constructing Histories from 2-4Transactions

Output:

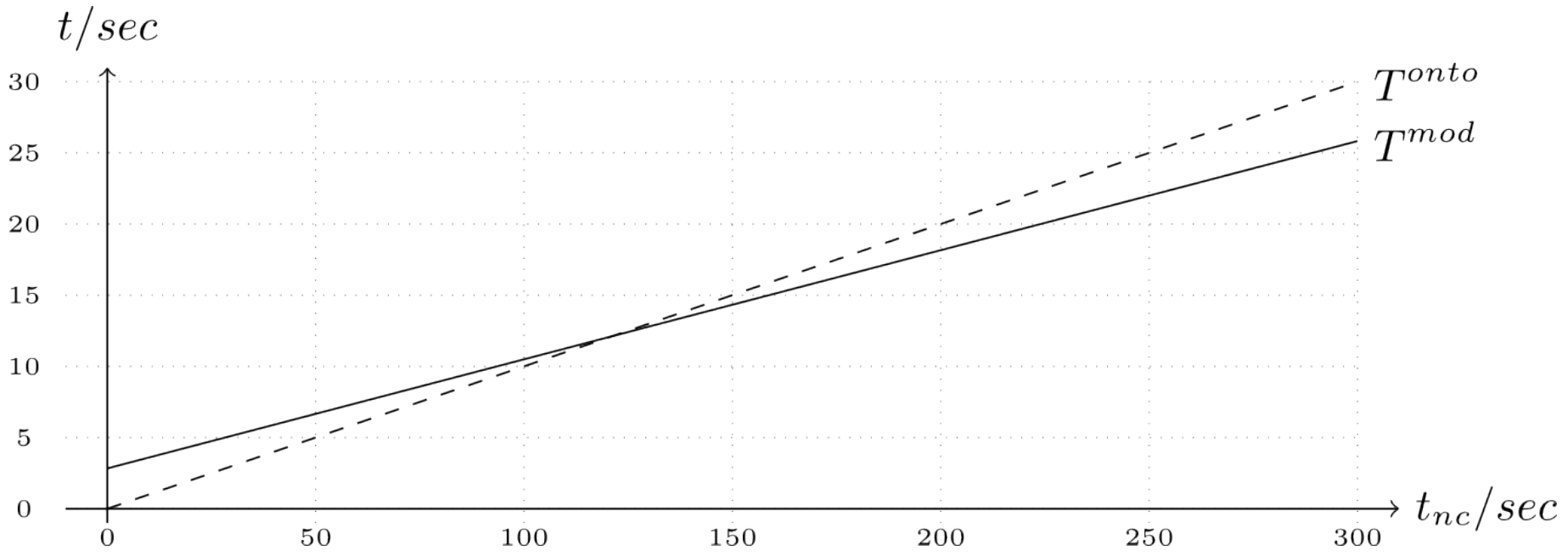
1200 Histories consisting of 24 – 96 atomic operations each

Evaluation Results

- 30% (~240) of the histories are serializable
  - Due to random generation of transactions
- Average serializable history 76,6% of the steps of serial execution
- In average 2.832 sec for a single lock calculation

- Linux Virtual Machine 8Gig Ram, 1 Core
  (Dual Xeon Hexacore, 2.9Ghz, 96 Gig Ram running ~20 VMs)

Results



- 10% of module calculation time for the module of the upcoming operation
- 90% to recalculate the lock

Incremental module calculation methods would lead to a downscale

Conclusion and Further Work

- Incremental module calculation
- Pre-computation strategies
- Real-world evaluation
- Integration into ontology development tool

Thank you for your attention.

See also [Stefan Scheglmann, Steffen Staab, Matthias Thimm, Gerd Gröner. Locking for Concurrent Transactions on Ontologies. In Proceedings of the 10th Extended Semantic Web Conference (ESWC'13). Montpellier, May, 2013.]