Provenance and Privacy

Susan B. Davidson

Zhuowei Bao, Sanjeev Khanna, Sudeepa Roy, Julia Stoyanovich
University of Pennsylvania

Sarah Cohen-Boulakia
Universite Paris-Sud

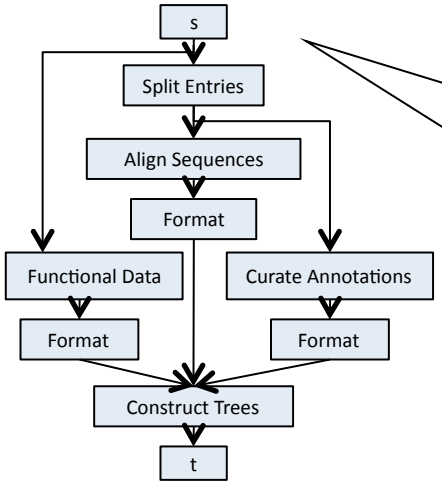
Tova Milo
Tel Aviv University

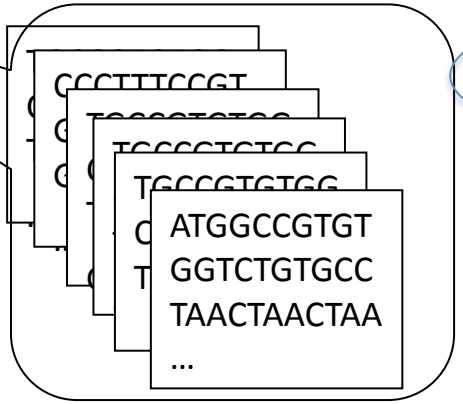
Desiderata from men (BPs)

- Discrete

Secure

Need for provenance

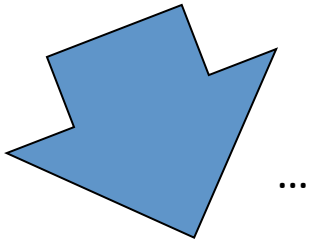




Need for privacy

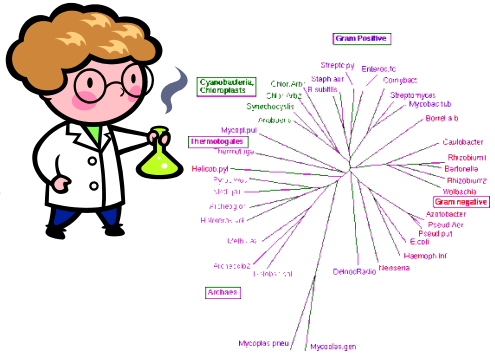
How has this tree been generated?

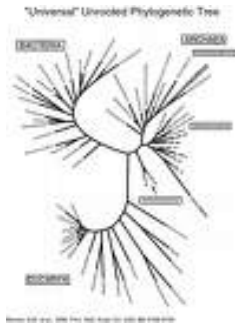
Which sequences have been used to produce this tree?

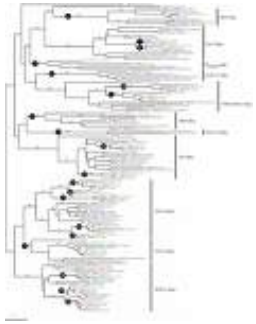


...

?







Biologist's workspace

Workflow Provenance Repositories

Current situation

- To enable sharing and reuse, repositories of workflow specifications are being created
  - e.g. myExperiment.org
  - Keyword search is used to find specifications of interest (tags at level of workflow)
- Several workflow systems are also storing provenance information
  - Module executions, input parameters, input/output data
  - **"Input-only"**

Vision

- "Workflow Provenance" repositories store specifications as well as executions (i.e. provenance)
  - Searchable
  - Queryable
- Searching/querying these repositories can be used to
  - Find/reuse workflows
  - Understand meaning of a workflow
  - Correct/debug erroneous specifications
  - See the downstream effect of "bad" data

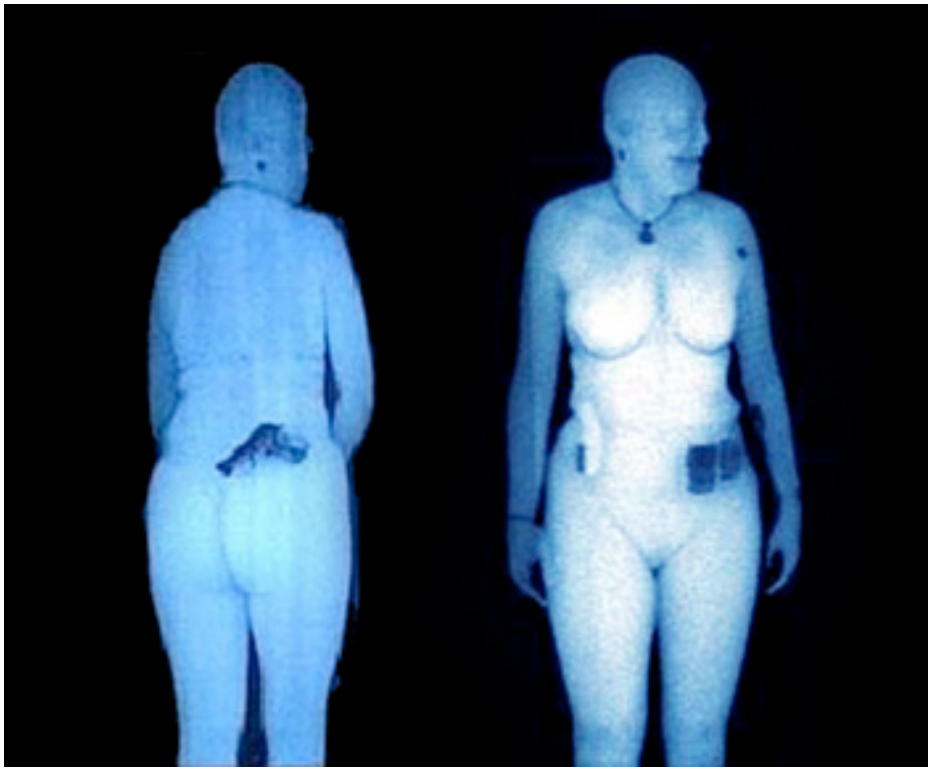
The Problem

- Owners/authors of workflows may wish to keep some of the provenance information private.
  - Intermediate data
  - Module behavior
  - Structure of the execution

➢ There is a tradeoff between the utility of provenance information and privacy guarantees.

➢ Search/query must respect privacy guarantees.

"You are better off designing in security and privacy… from the start, rather than trying to add them later."



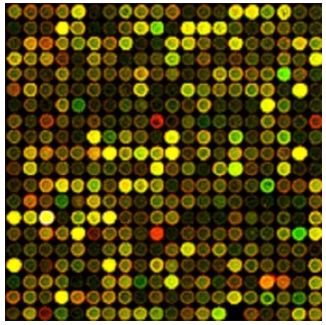
Privacy Concerns in Data-Oriented Workflows

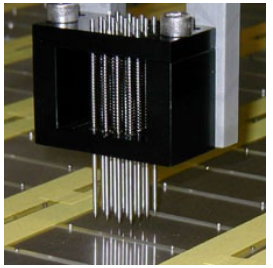
Privacy and Workflow Provenance

- Privacy concerns are tied to the components of workflow provenance
  - Data that flows on edges
  - Modules that implement functions
  - Structure of provenance dependencies: "connections" between data and other data, or between data and module executions

Example 1: Data Privacy

Microarray data obtained from the experiment



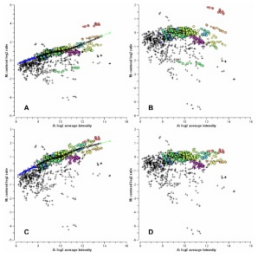


Robots are used to perform microarray analysis

Data must be normalized to be interpreted correctly

Normalization data should be kept secret

Microarray companies provide normalization methods

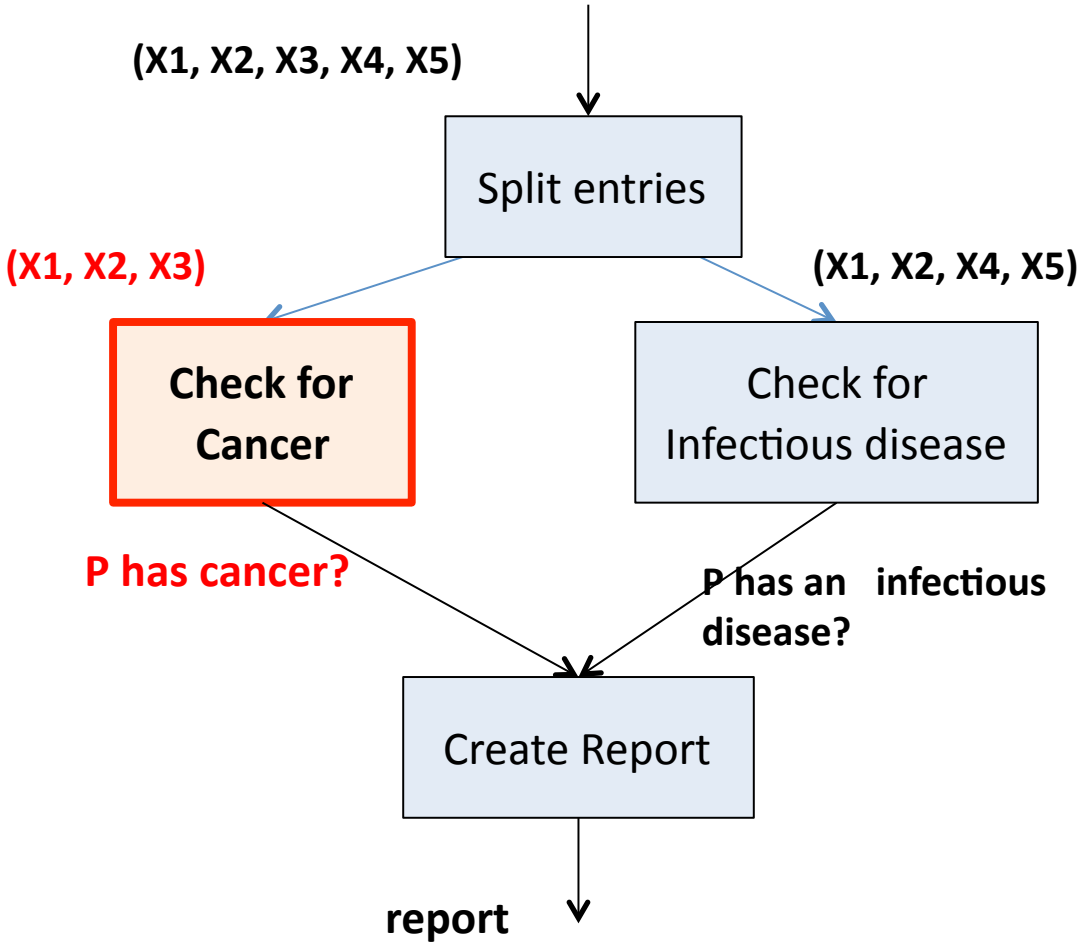


Normalized data

Data from other groups is used in normalization

Example 2: Module Privacy

Patient record: Gender, smoking habits,
Familial environment,
blood pressure, blood test report, …

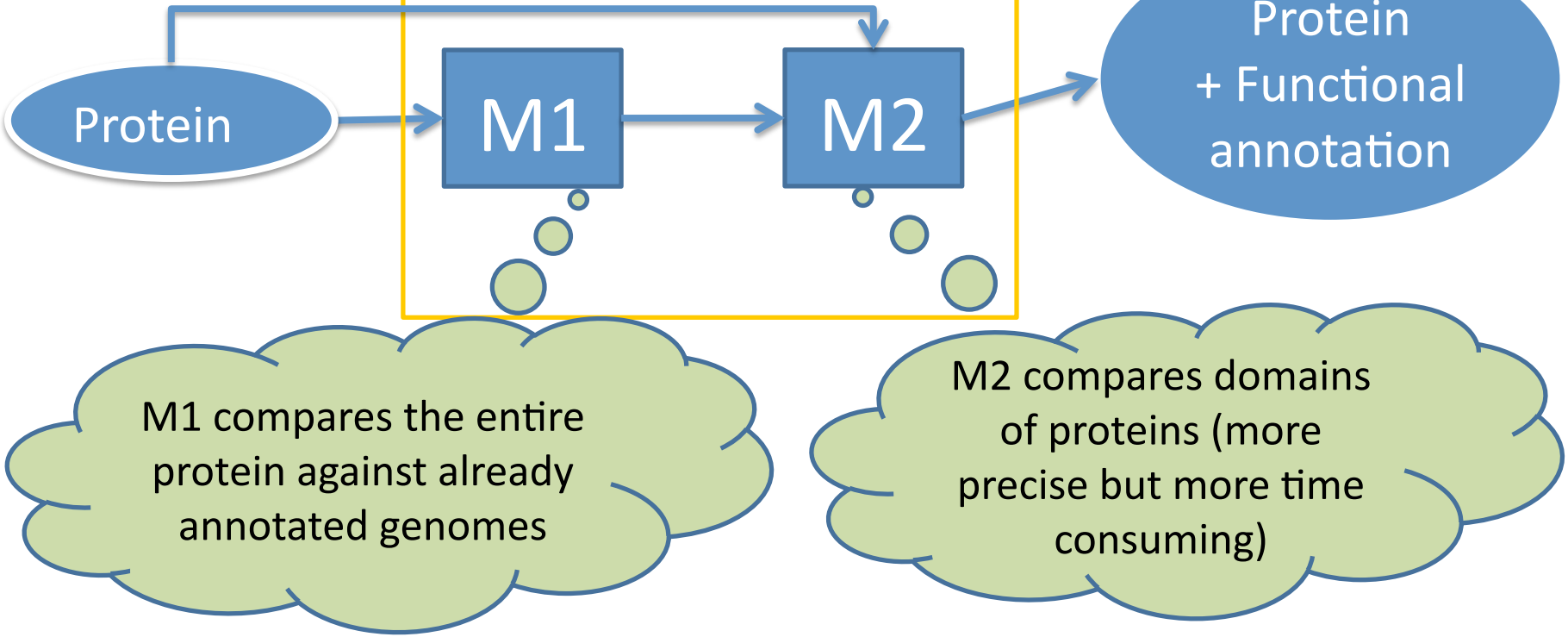


Module functionality should be kept secret

From patient's standpoint: output should not be guessed given input data values

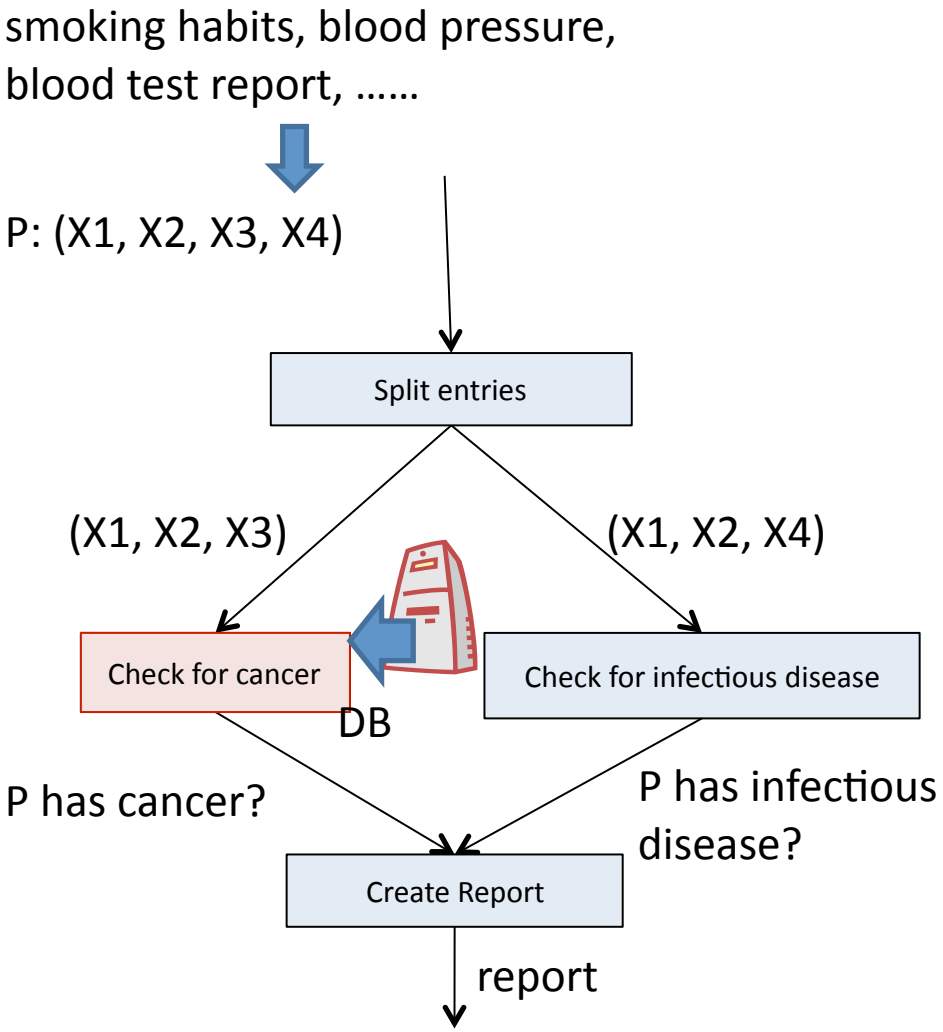
From module owner's standpoint: no one should be able to simulate the module and use it elsewhere.

Example 3: Structural Privacy



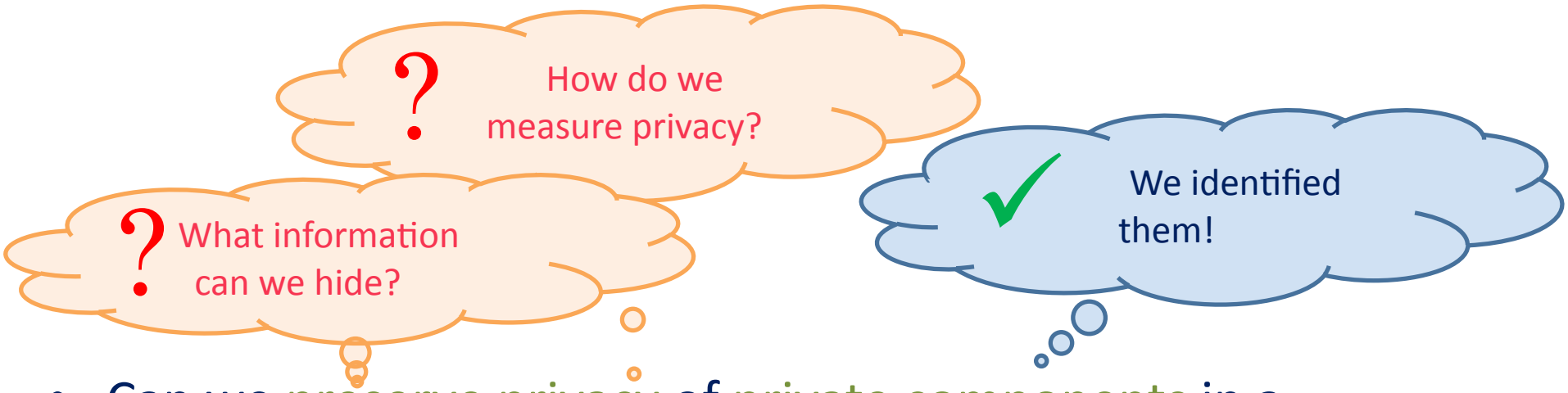
Relationships between certain data/module pairs should be kept secret

Privacy concerns at a glance

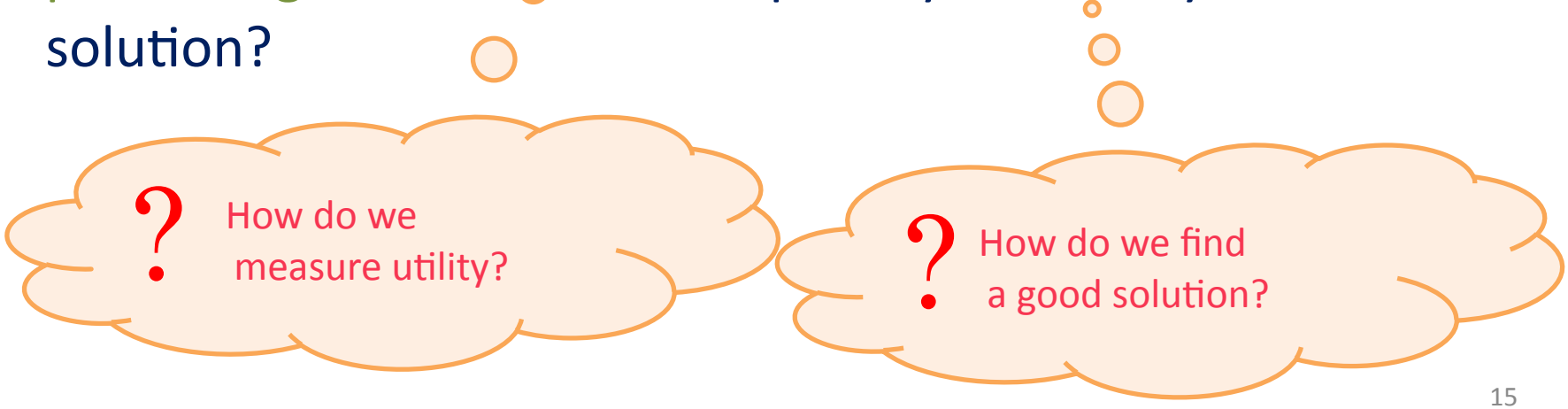


- Data Privacy
  - Data items are private
- Module Privacy
  - Module functionality is private (x, f(x))
- Structural Privacy
  - Execution paths between certain data is private

The questions we need to answer...



- Can we preserve privacy of private components in a workflow and maximize utility w.r.t. provenance queries with provable guarantees on both privacy and utility of the solution?



Module Privacy (a hint)

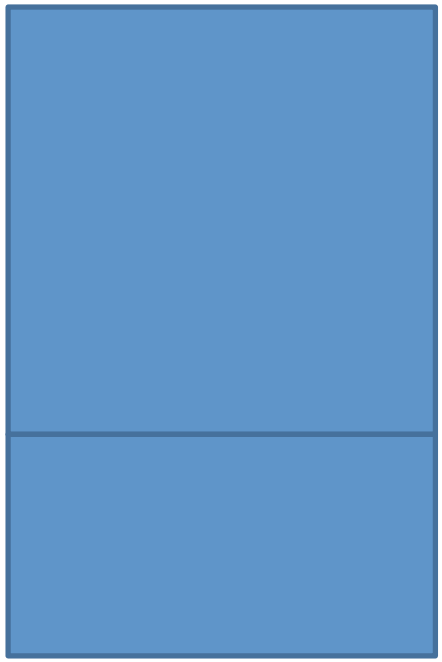
Roy et al, PODS 2011

- A module m = a function
- For every input x to m, m(x) value should not be revealed
  - Enough equivalent possible m(x) values w.r.t. visible information

Module Privacy (a hint)

- A module m = a function
- For every input x to m, m(x) value should not be revealed
  - Enough equivalent possible m(x) values w.r.t. visible information

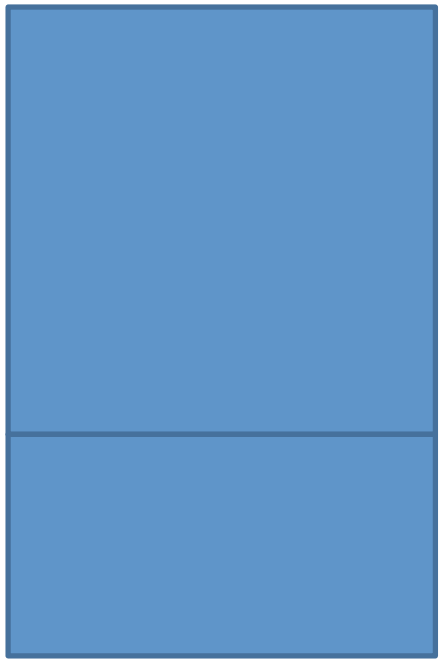
- There is a knife, a fork and a spoon in this figure



Module Privacy (a hint)

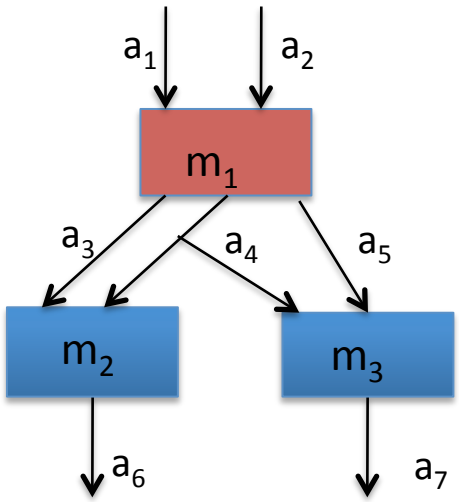
- A module m = a function
- For every input x to m, m(x) value should not be revealed
  - Enough equivalent possible m(x) values w.r.t. visible information

- There is a knife, a fork and a spoon in this figure



Γ-privacy

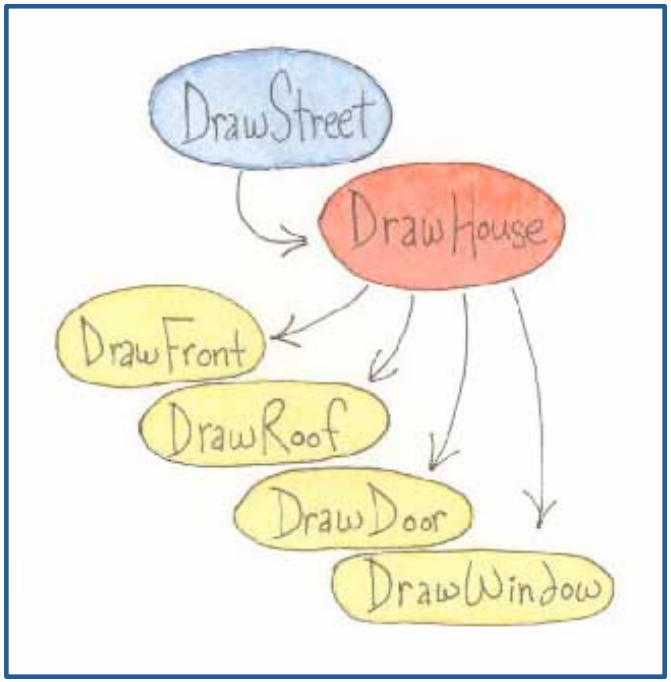
- A module m is Γ-private iff for every input x the actual value of m(x) is indistinguishable from Γ-1 other possible values wrt the visible data.
  - **Example:** Hiding a_2 and a_4 in the provenance table for m_1 guarantees 4-privacy. E.g. $m_1(0,0)$ could be (0,0,1), (0,1,1), (1,0,0) or (1,1,0).



Input		Output		
a_1	a_2	a_3	a_4	a_5
0	0	0	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	0	1

Input		Output		
a_1	a_2	a_3	a_4	a_5
0		0		1
0		1		0
1		1		0
1		1		1

Hierarchical Workflow Model



Composite Modules

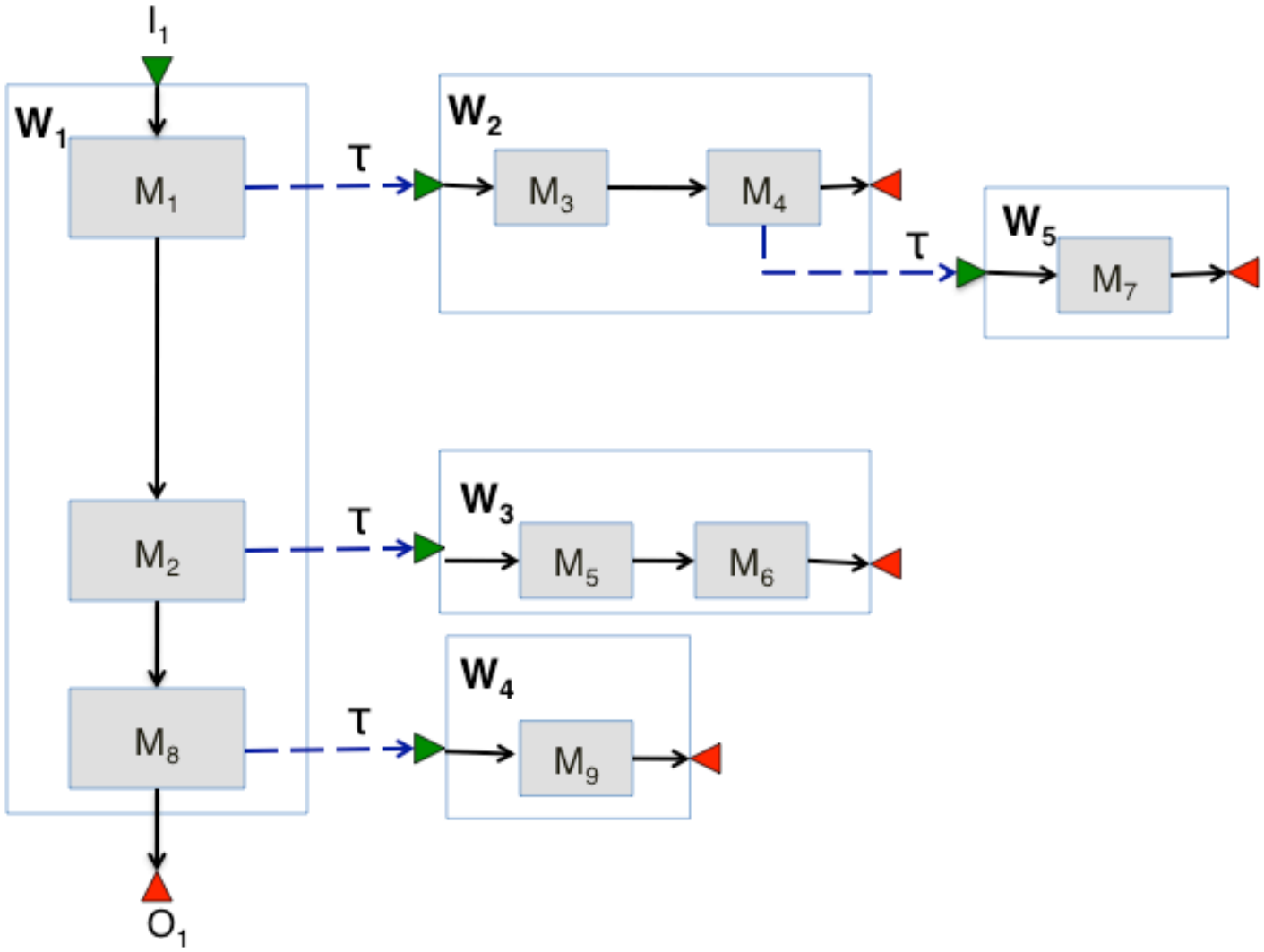
- Composite modules encapsulate subworkflows
  - Extensively used in workflow design to enable reuse and sharing (top-down)
  - Also used to hiding portions of provenance to focus on "relevant" modules (bottom-up)
    - Biton et al, VLDB 2007, ICDE2008, ICDT2009; Sun et al, PVLDB 2009, SIGMOD2009
- Whether developed top-down or bottom up, composite modules can be used to create views of workflows or their provenance.

➢ Yields a hierarchical workflow model.

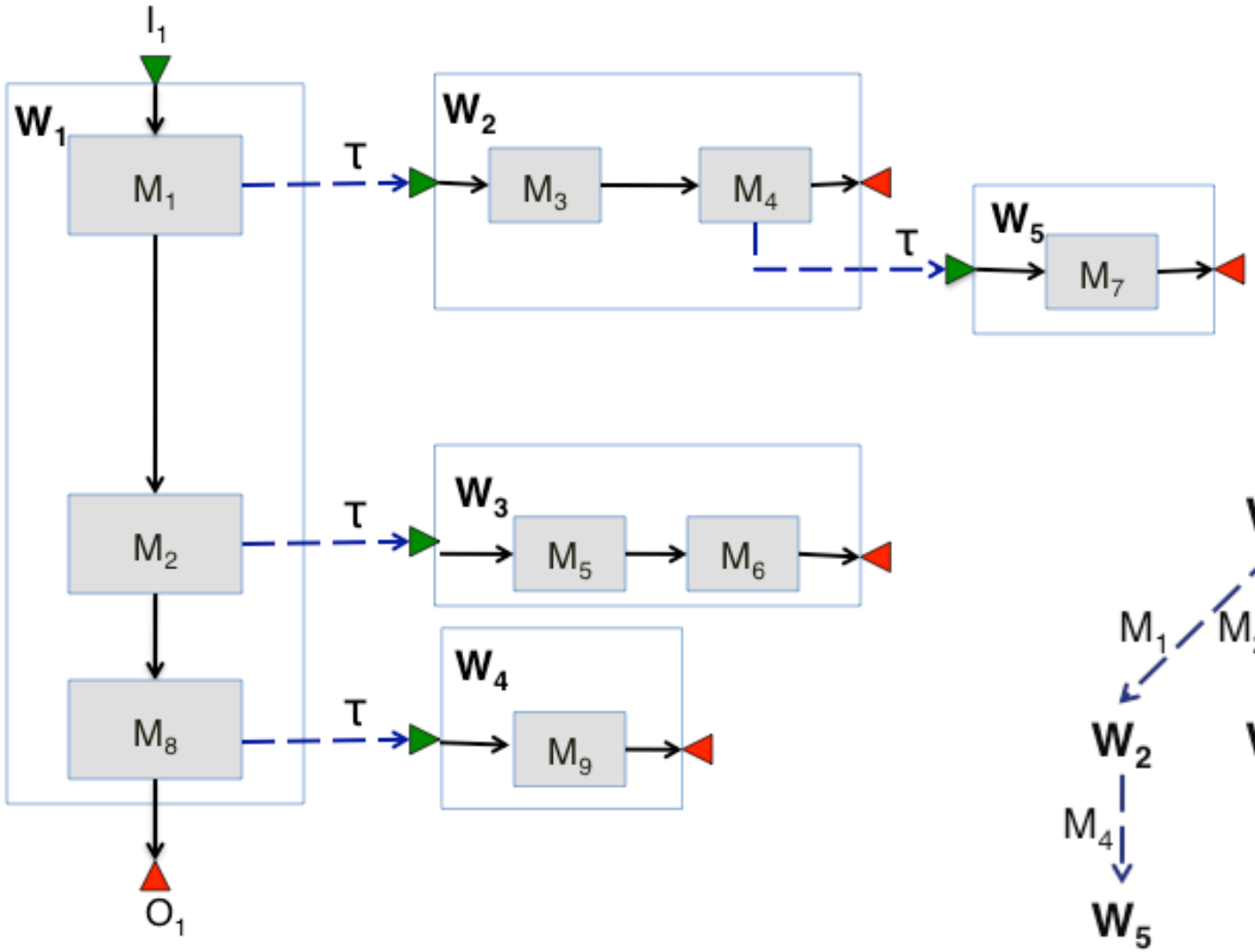
Workflow model, revisited

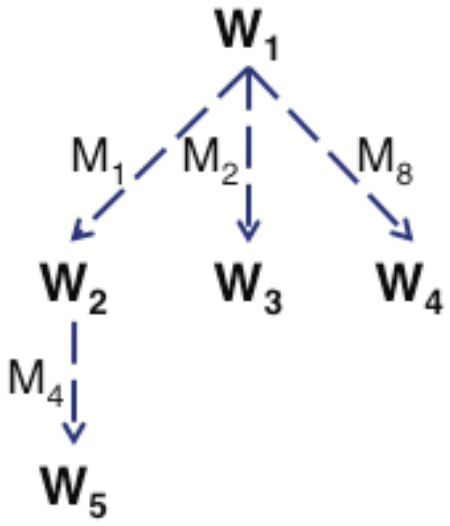
- A simple workflow is a connected DAG whose nodes model modules and edges model potential dataflow between modules.
- A (hierarchical) workflow is a pair (W, τ) where W is a finite set of simple workflows and τ is a (partial) expansion function that maps some of the modules to simple workflows in W.
- Expansion edges can be used to define an expansion hierarchy

Hierarchical workflow, example

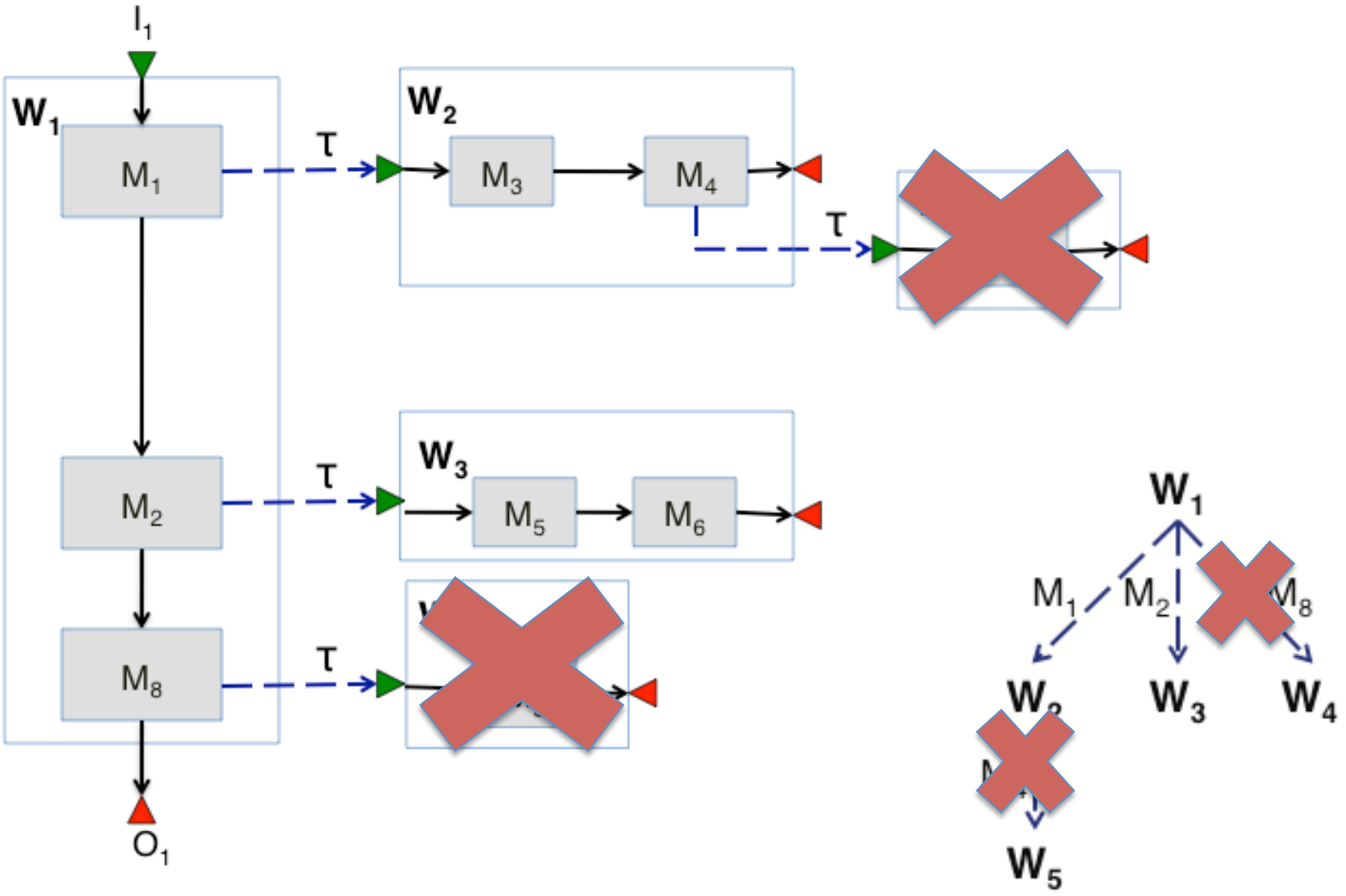


Expansion hierarchy, example





View: Prefix of hierarchy



Views are useful…

- At the specification level, views can be used to control what is seen of module descriptions or the expansion to a subworkflow.
- View can also be projected to the execution level (provenance) to control what data is seen, hide structural information, or hide inferred module behavior

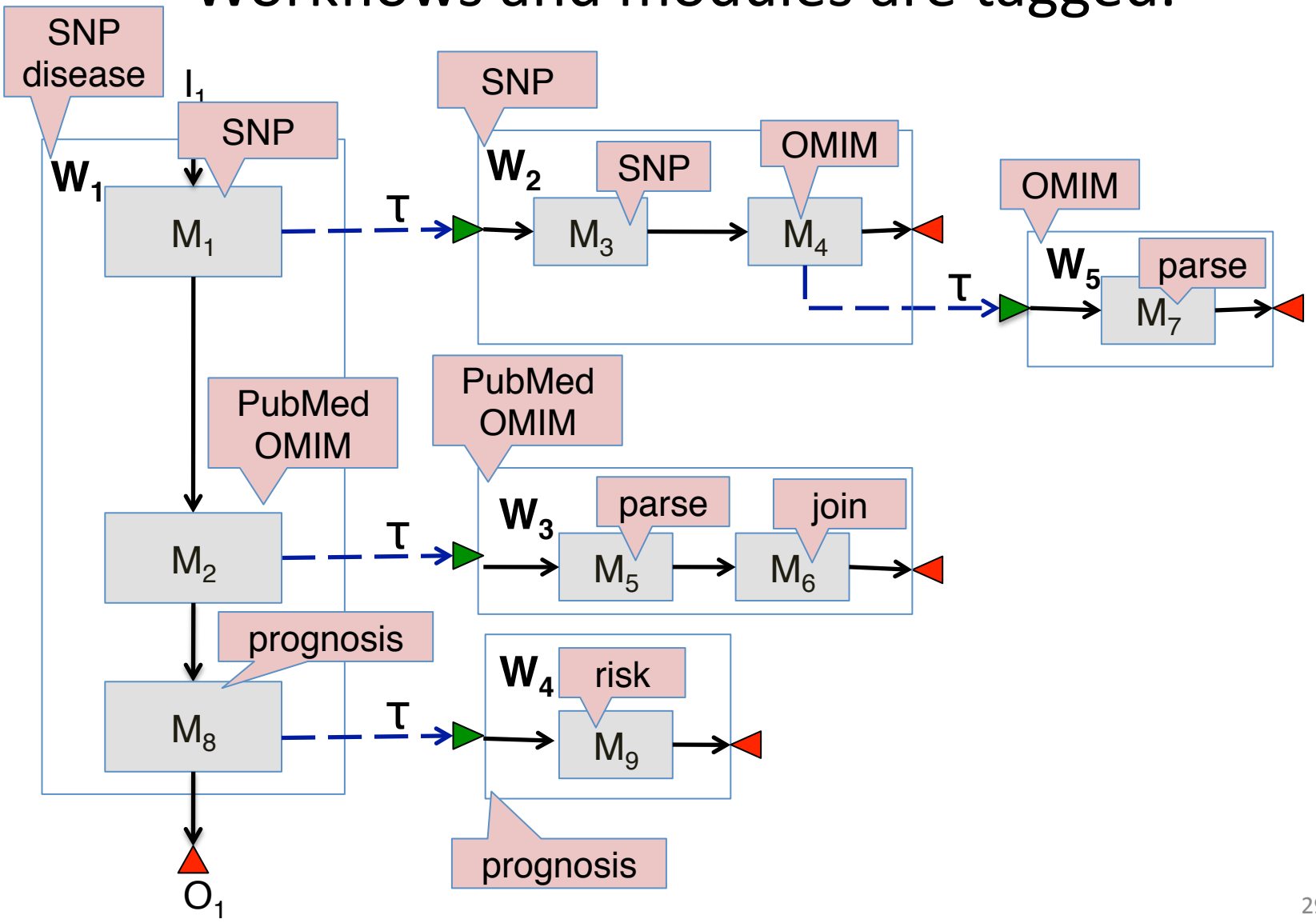
Privacy-aware Search and Query

The Vision, recapped

- Workflow Provenance repositories will store specifications as well as executions (i.e. provenance)
  - Searchable, queryable
- Query results must respect privacy guarantees.
  - Data, module, structure.

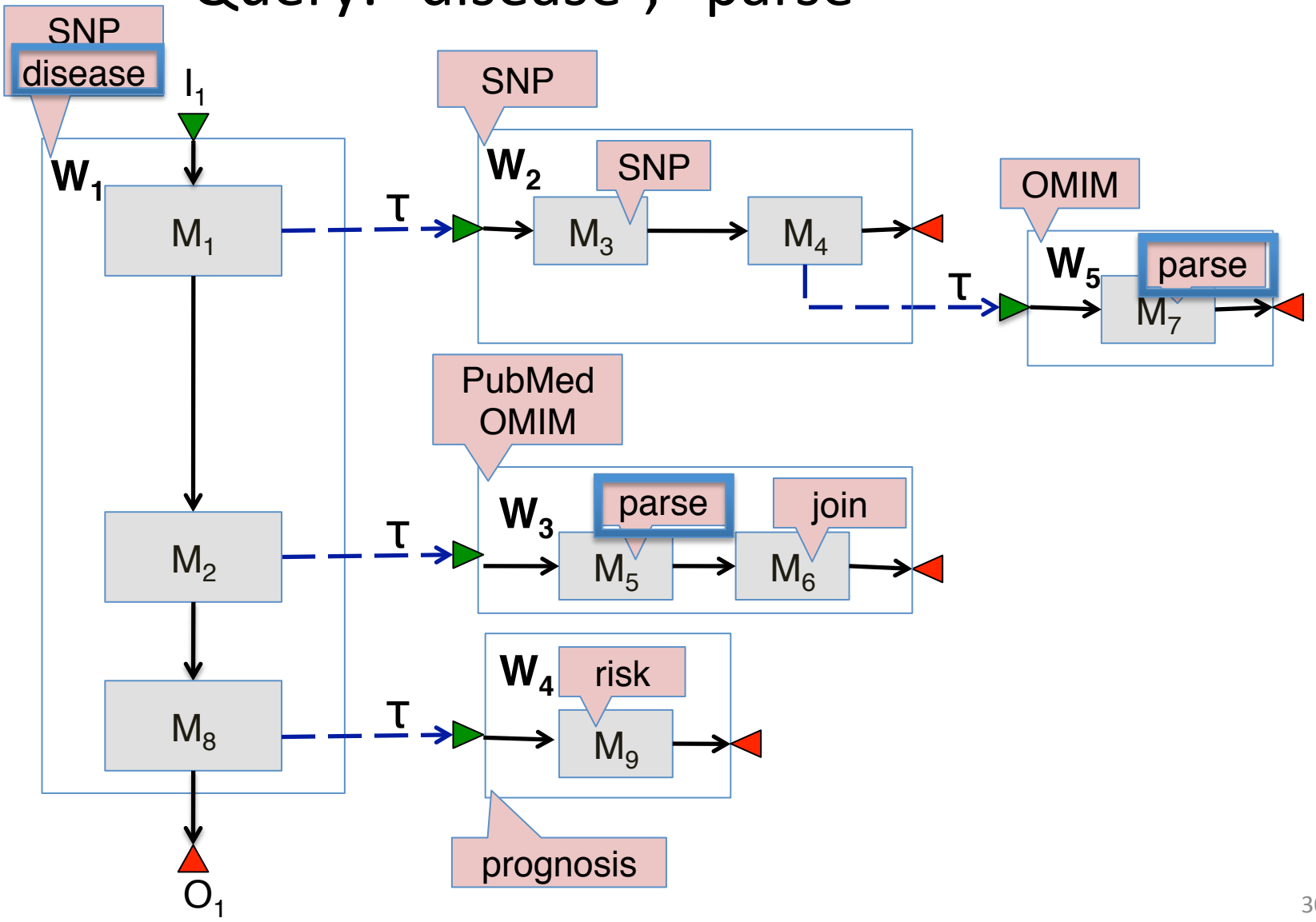
Search

Workflows and modules are tagged.



Search

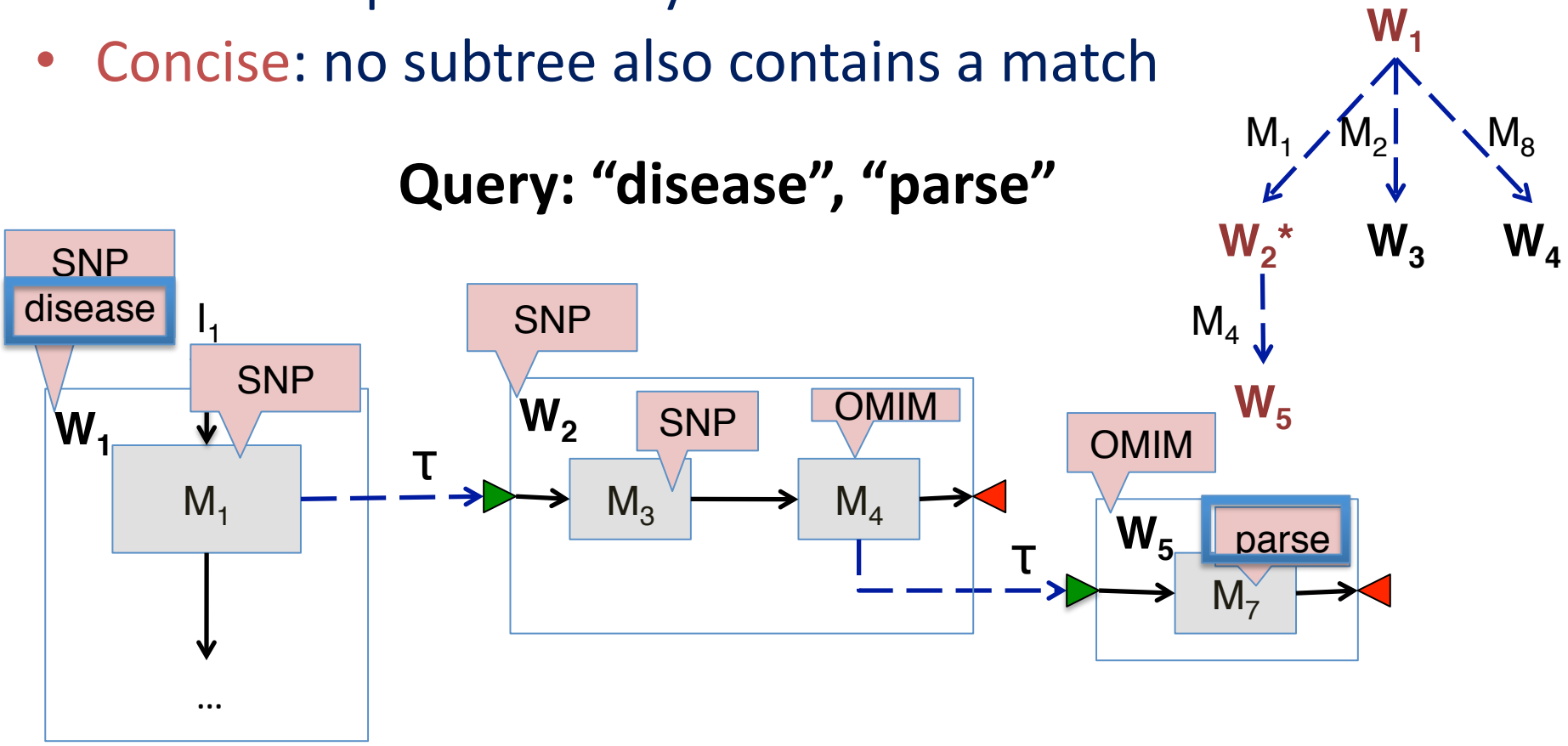
Query: “disease”, “parse”



Search result

"WISE: Searching workflow hierarchies": Liu et al, VLDB 2007

- Informative: shows the expansion and dataflow relationships necessary to understand the match
- Concise: no subtree also contains a match



Access Control Specification

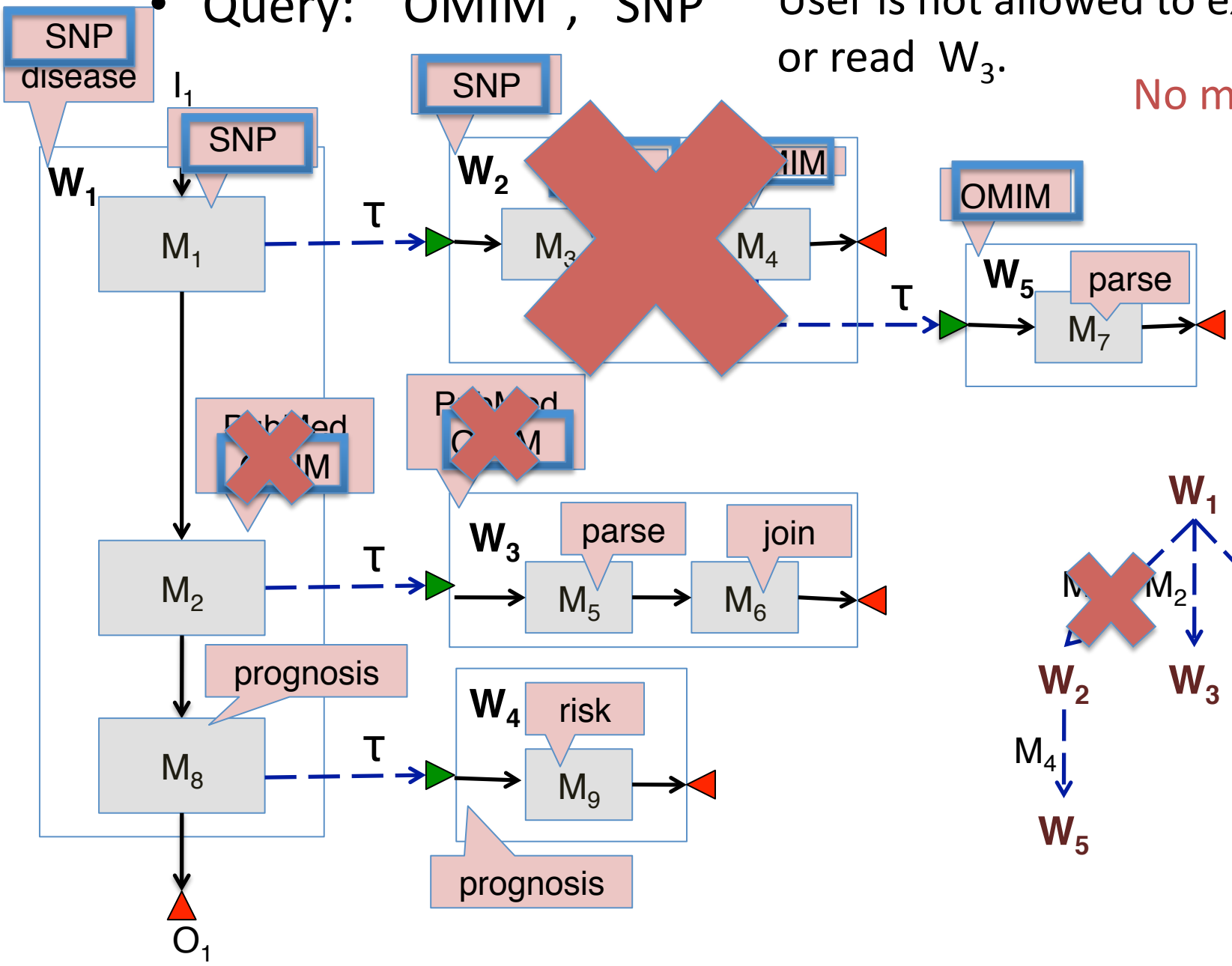
- Each module/workflow S has two actions
  - Read: authorized users can access keywords of S
  - Expand: authorized users can see the structure of S.
- The expand privileges for a user can be used to "trim" the expansion hierarchy and create an access view.
- The user's access view and read privileges can be used to control what is returned in a search.
- Access controlled repository – same privileges on a module and on a workflow to which it expands.

Access Controlled Search

- Query: "OMIM", "SNP"

User is not allowed to expand W_2 or read W_3.

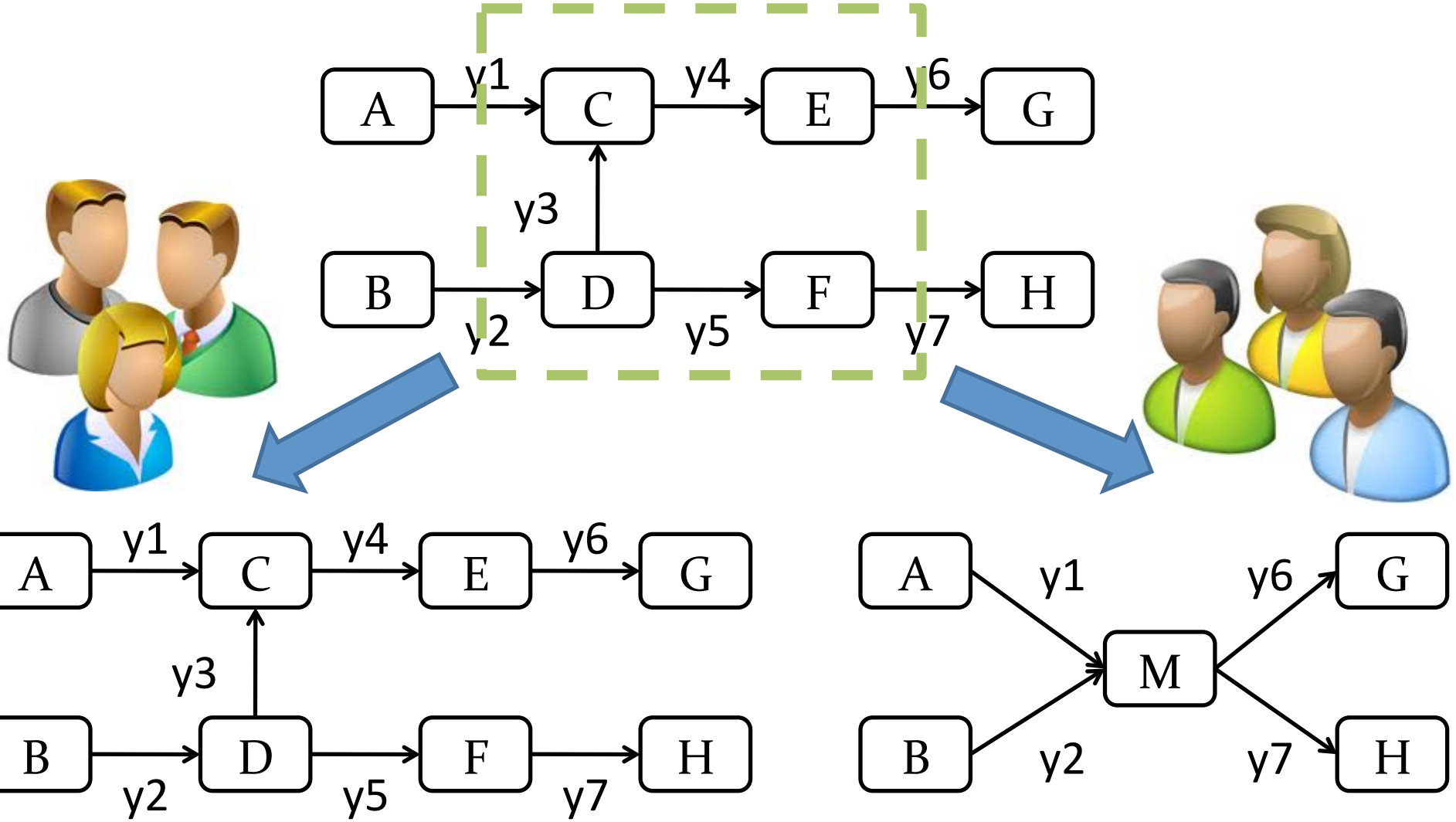
No match

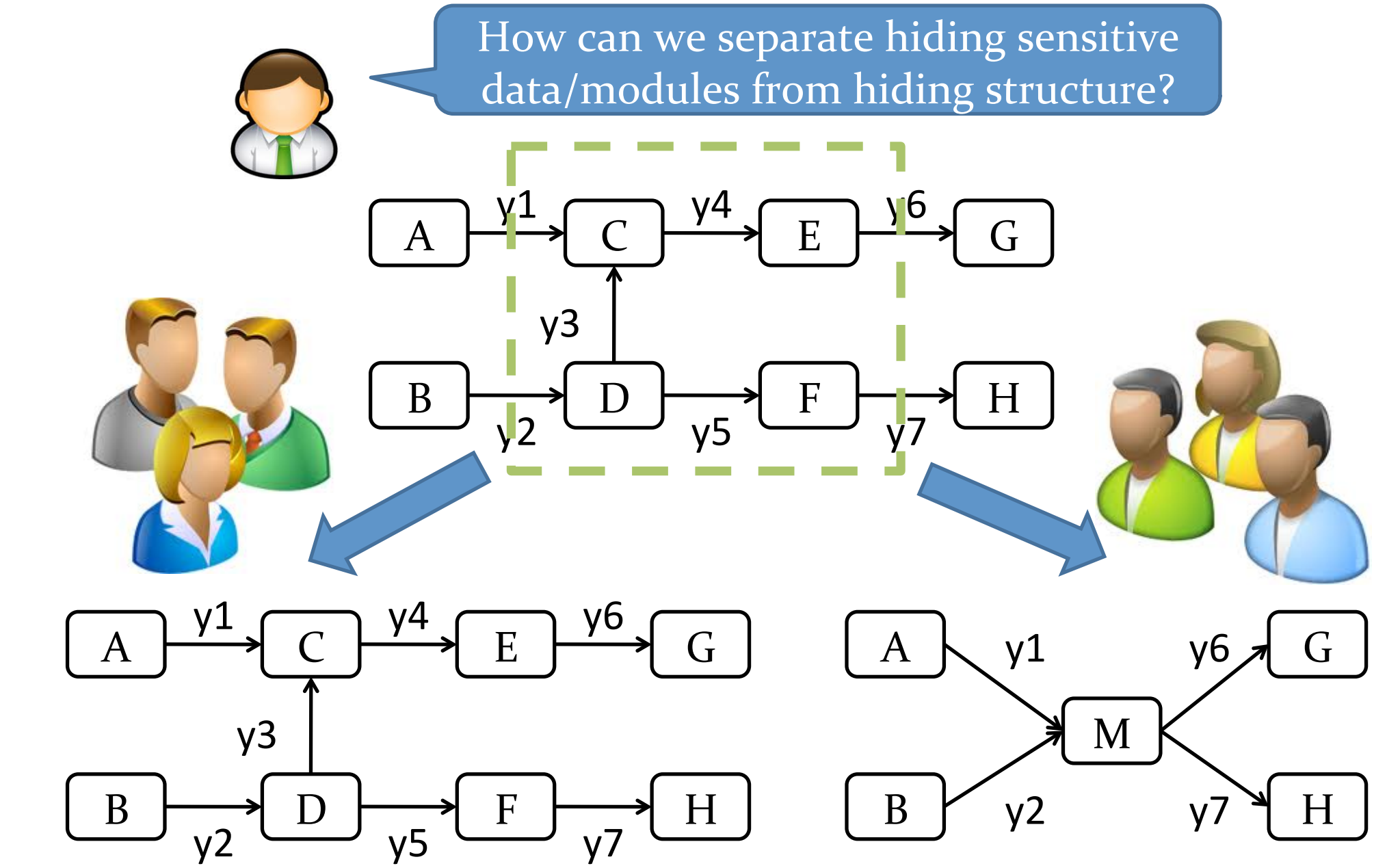


What about structural privacy?

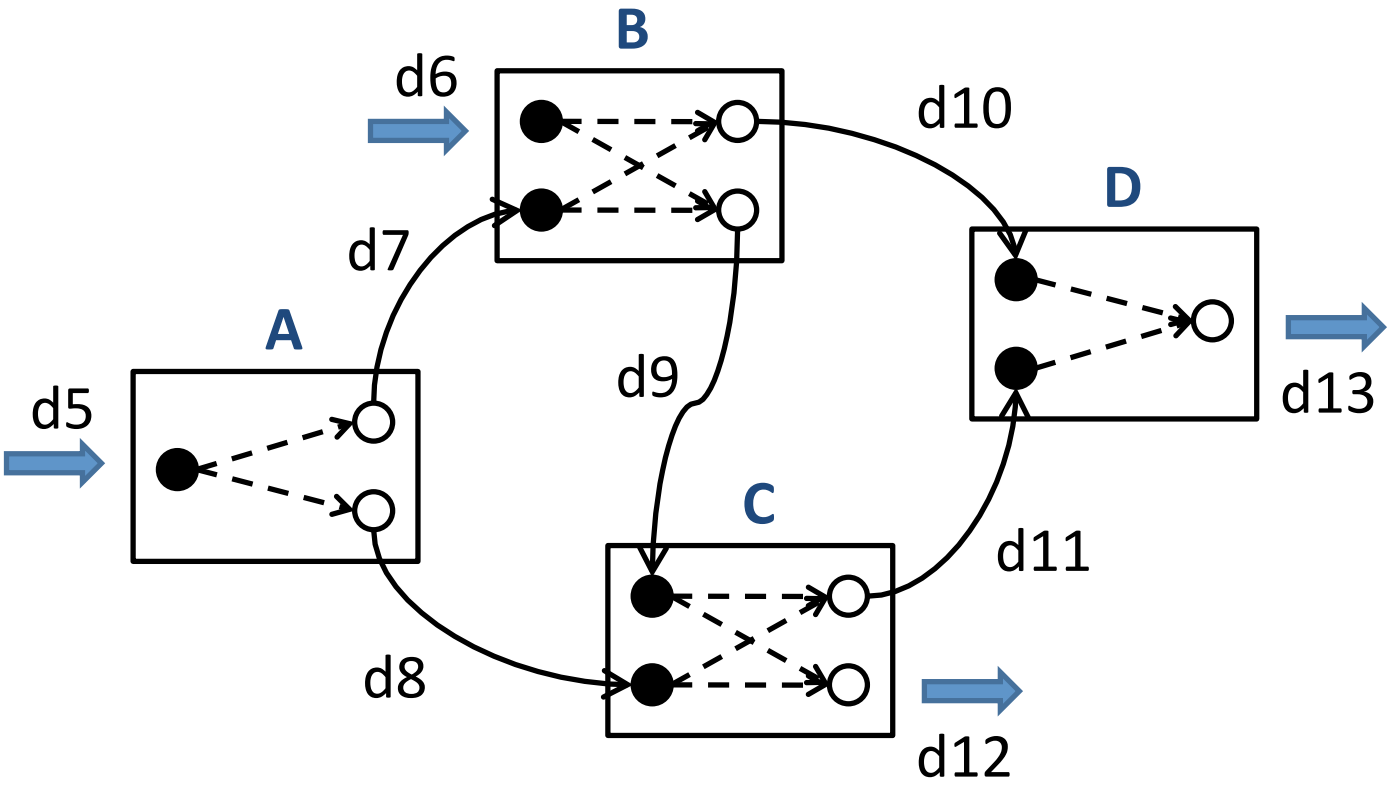
Poster with Bau and Milo

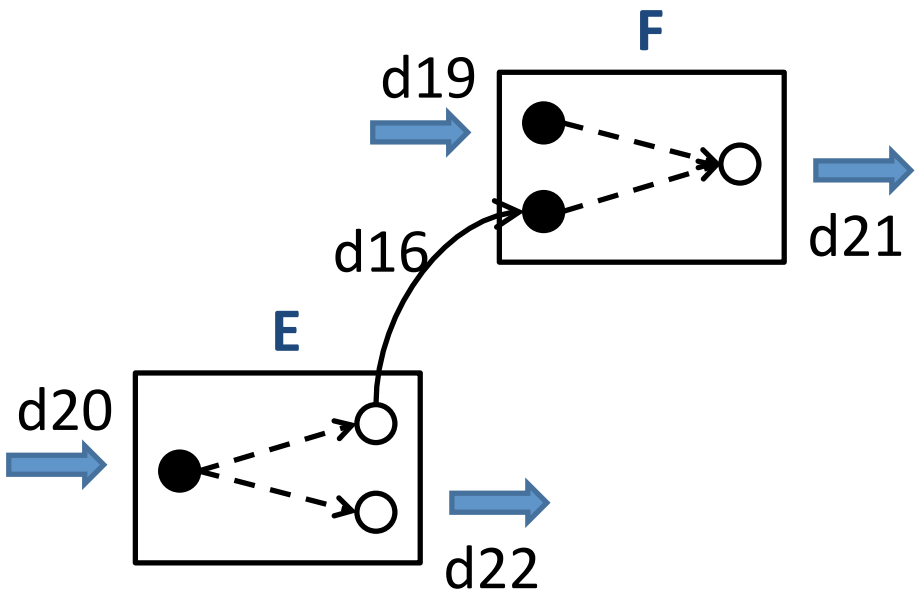
Access (Security) Views



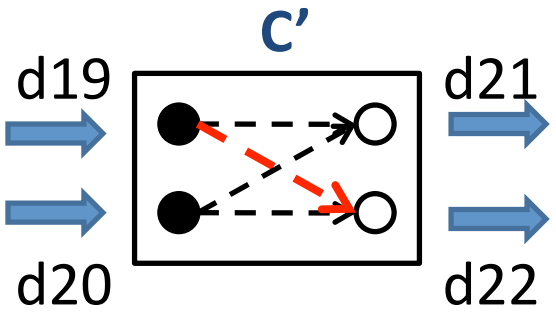

How can we separate hiding sensitive data/modules from hiding structure?
A
C
E
G
B
D
F
H
y1
y4
y6
y3
y2
y5
y7
M

Dependency and Data Edges





Simple workflow



Composite module that expands to the simple workflow

Research Challenges

- "Workflow Provenance" repositories will store specifications as well as executions (i.e. provenance)
  - ✓ Searchable, queryable, privacy preserving
- Formalizing privacy notions
  - **Data privacy:** Hiding a data value may not be enough – how much is revealed from the displayed data values?
  - **Module privacy:** how to handle workflows with both private and public modules?
  - **Structural privacy:** What techniques should be used? What are the desired guarantees?
  - Can we use differential privacy?
- Search: efficiently identifying data that users can access
  - Users may have different privileges, yielding many different "access views".
- What is an appropriate provenance query language? How does access control interact?

Research Challenges, cont.

- How to express security policies and ensure they are “obeyed”
- There is also related work on secure provenance, i.e. detecting and protecting against provenance tampering
- …

Session Papers

- "A Framework for Policies over Provenance" (Tyrone)
  - Specify access control and redaction policies which transform provenance graph to hide sensitive information
- "Tracking Emigrant Data via Transient Provenance" (Stephanie)
  - Best security policies can be compromised by trusted party with malicious intent
  - Ghost objects track when data leaves system
- "One of These Records Is Not Like the Others" (Carrie)
  - Propose various techniques (crypto, consistency checks) to detect and correct errors in provenance
  - Consistency checks can be thwarted by rogue generator examining provenance records to supply info for new record: may need to secure provenance record for others than creator.
- "A Fine-Grained Wf Model with Provenance-Aware Security Views"
  - Specify for each module (atomic or composite) the input/output dependencies. Users are given a view at which level they can see the workflow provenance. This can be used for data/module/structural privacy.

Acknowledgments

- Members of the PennDB group
  - Sanjeev Khanna
  - Sudeepa Roy
  - Julia Stoyanovich
  - Val Tannen
- Friend of PennDB and Tel Aviv faculty
  - Tova Milo
- PennDB alum and ASU faculty
  - Yi Chen
- Bioinformatics collaborators
  - Sarah Cohen-Boulakia

This work was supported in part by NSF IIS-0803524, IIS-0629846, IIS-0915438, CCF-0635084, and IIS-0904314; NSF CAREER award IIS-0845647; and CRA 0937060