Dependency graphs and program transformations in the computationally sound security analysis of cryptographic protocols

Peeter Laud
peeter.laud@ut.ee
http://www.cs.ut.ee/~peeter_l
Tartu University & Cybernetica AS, sometimes Inst. of Cybernetics
(joint work with Ilja Tšahhirov)

Dependency graphs

- Directed graph, nodes are labeled with operations.
  - The label of a node determines its in-degree.
  - Incoming edges are (usually) ordered.
- Nodes of a DG compute values, purely functionally.
- Edges describe where the values are used for further computations.
- Special nodes are used to bring inputs to the system.
  - . . . incl. randomness;
  - . . . incl. activation information.
- Also, there are special nodes denoting outputs.

Security definitions of cryptoprimitives

- Cryptographic primitives — encryption, signatures, MACs, hash functions, etc.
- A (computational) security definition states that the adversary will be unsuccessful in a certain environment containing that primitive.
- Often can be stated as:
  - The adversary, interacting with one of the structures S_0 or S_1, cannot tell which one it is talking to.
  - Structure — a program (fragment) / a subroutine.
  - S_1 corresponds to the "real use" of the primitive.
  - S_0 is a kind of idealized behaviour.
- The following operation does not (distinguishably) change the observable behaviour of a program P:
  - Locate (an instance of) the subroutine S_b in P;
  - Replace it with S_{1-b}.

Cryptographic protocols

- Small, distributed programs.
- Sequence-of-games method of analysis:
  - Search the protocol text for an instance of the structure S_1 for some cryptographic primitive it's using.
  - Replace that instance with S_0.
  - Do the previous steps as long as possible.
  - Analyse the resulting protocol instead, using some "non-cryptographic" method.
    - E.g. secrecy — check that there are no references to secret values.
- Searches for the instances of S_1 are easy if the protocol is represented as a dependency graph.

Example protocol

- A wants to send the secret M to B.
- S is a trusted server.

$$
\begin{aligned}
A &\longrightarrow B: \; A, B, \{N_A\}_{K_{AS}} \\
B &\longrightarrow S: \; A, B, \{N_A\}_{K_{AS}}, \{N_B\}_{K_{BS}} \\
S &\longrightarrow A: \; \{K_{AB}, N_A\}_{K_{AS}} \\
S &\longrightarrow B: \; \{K_{AB}, N_B\}_{K_{BS}} \\
A &\longrightarrow B: \; \{M\}_{K_{AB}}
\end{aligned}
$$

Example protocol

- A wants to send the secret M to B.
- S is a trusted server.

$$
\begin{aligned}
A &\longrightarrow B: \; A, B, \{N_A\}_{K_{AS}} \\
B &\longrightarrow S: \; A, B, \{N_A\}_{K_{AS}}, \{N_B\}_{K_{BS}} \\
S &\longrightarrow A: \; \{K_{AB}, N_A\}_{K_{AS}} \\
S &\longrightarrow B: \; \{K_{AB}, N_B\}_{K_{BS}} \\
A &\longrightarrow B: \; \{M\}_{K_{AB}}
\end{aligned}
$$

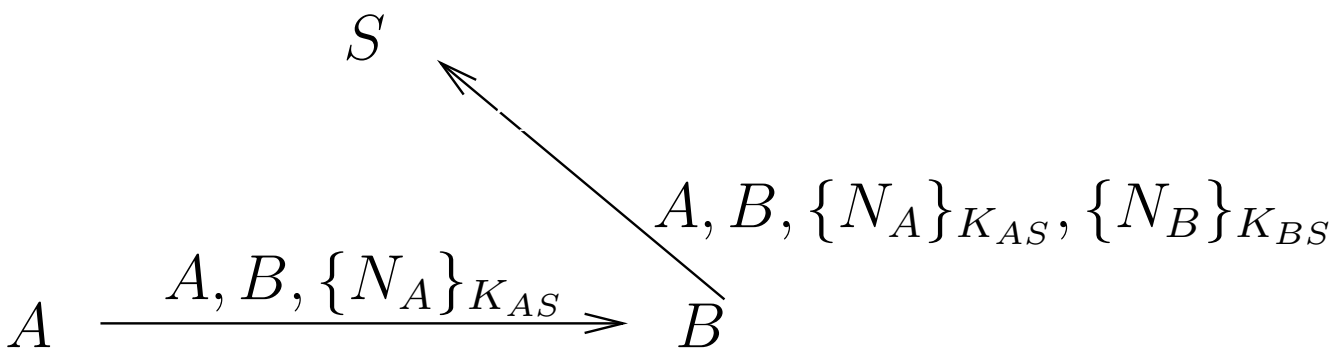
$$S$$

$$A \xrightarrow{A, B, \{N_A\}_{K_{AS}}} B$$

Example protocol

- A wants to send the secret M to B.
- S is a trusted server.

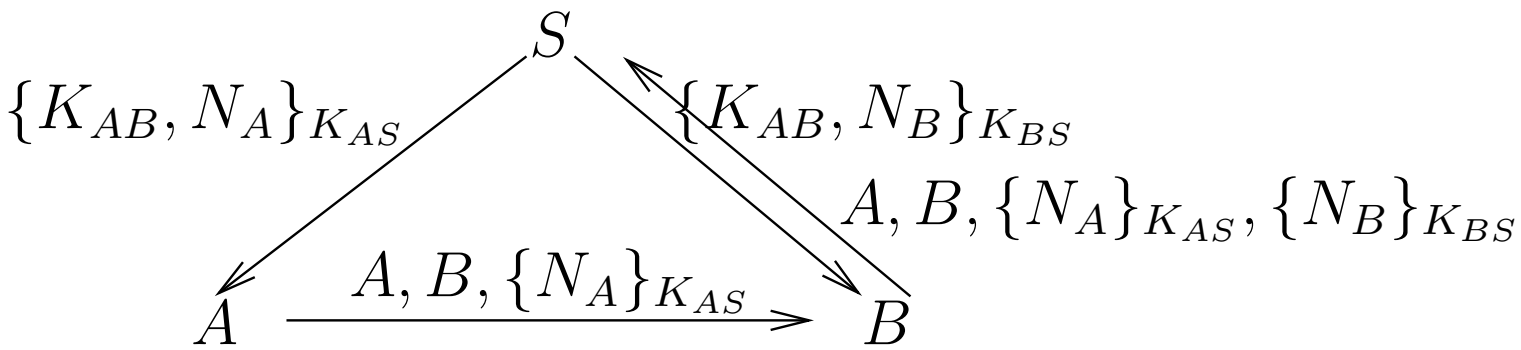
$$
\begin{aligned}
A &\longrightarrow B: \; A, B, \{N_A\}_{K_{AS}} \\
B &\longrightarrow S: \; A, B, \{N_A\}_{K_{AS}}, \{N_B\}_{K_{BS}} \\
S &\longrightarrow A: \; \{K_{AB}, N_A\}_{K_{AS}} \\
S &\longrightarrow B: \; \{K_{AB}, N_B\}_{K_{BS}} \\
A &\longrightarrow B: \; \{M\}_{K_{AB}}
\end{aligned}
$$



Example protocol

- A wants to send the secret M to B.
- S is a trusted server.

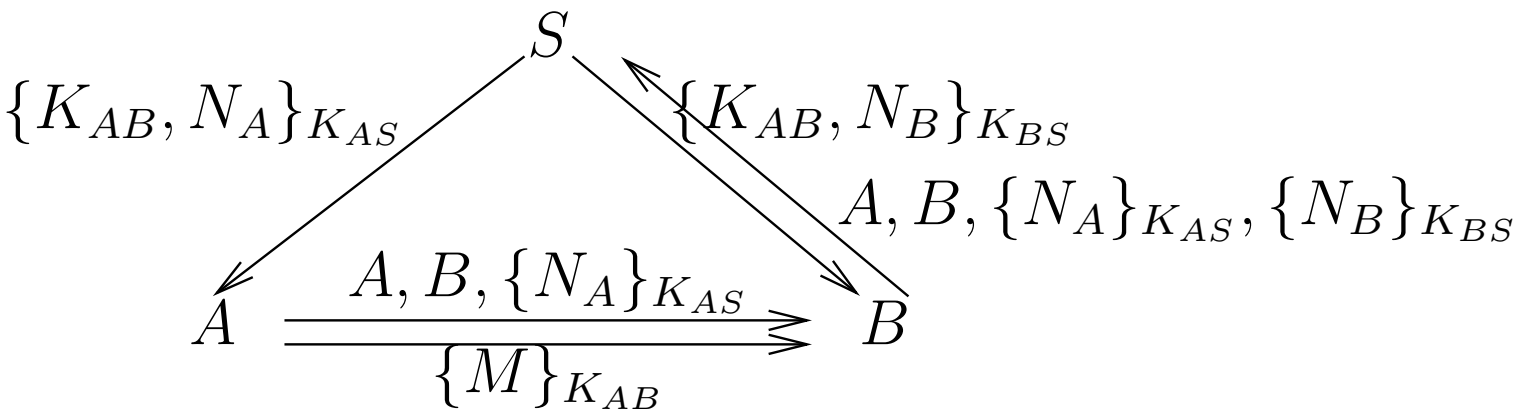
$$
\begin{aligned}
A &\longrightarrow B: \; A, B, \{N_A\}_{K_{AS}} \\
B &\longrightarrow S: \; A, B, \{N_A\}_{K_{AS}}, \{N_B\}_{K_{BS}} \\
S &\longrightarrow A: \; \{K_{AB}, N_A\}_{K_{AS}} \\
S &\longrightarrow B: \; \{K_{AB}, N_B\}_{K_{BS}} \\
A &\longrightarrow B: \; \{M\}_{K_{AB}}
\end{aligned}
$$



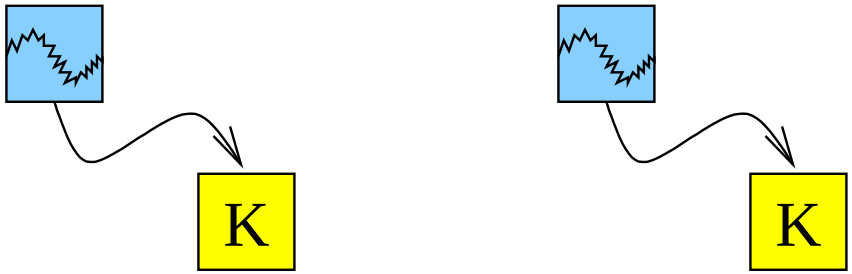
Example protocol

- A wants to send the secret M to B.
- S is a trusted server.

$$
\begin{aligned}
A &\longrightarrow B: \; A, B, \{N_A\}_{K_{AS}} \\
B &\longrightarrow S: \; A, B, \{N_A\}_{K_{AS}}, \{N_B\}_{K_{BS}} \\
S &\longrightarrow A: \; \{K_{AB}, N_A\}_{K_{AS}} \\
S &\longrightarrow B: \; \{K_{AB}, N_B\}_{K_{BS}} \\
A &\longrightarrow B: \; \{M\}_{K_{AB}}
\end{aligned}
$$

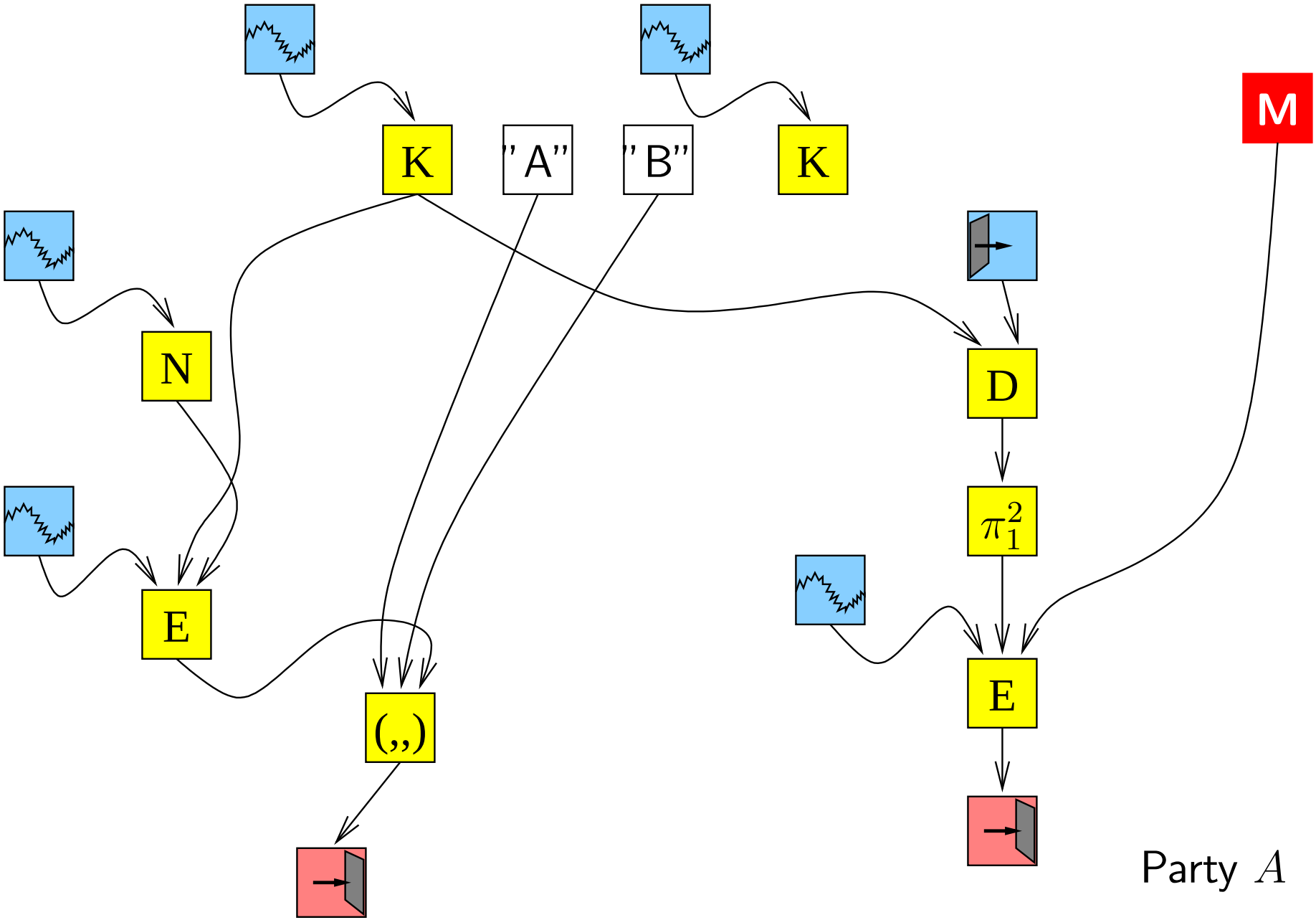


A protocol

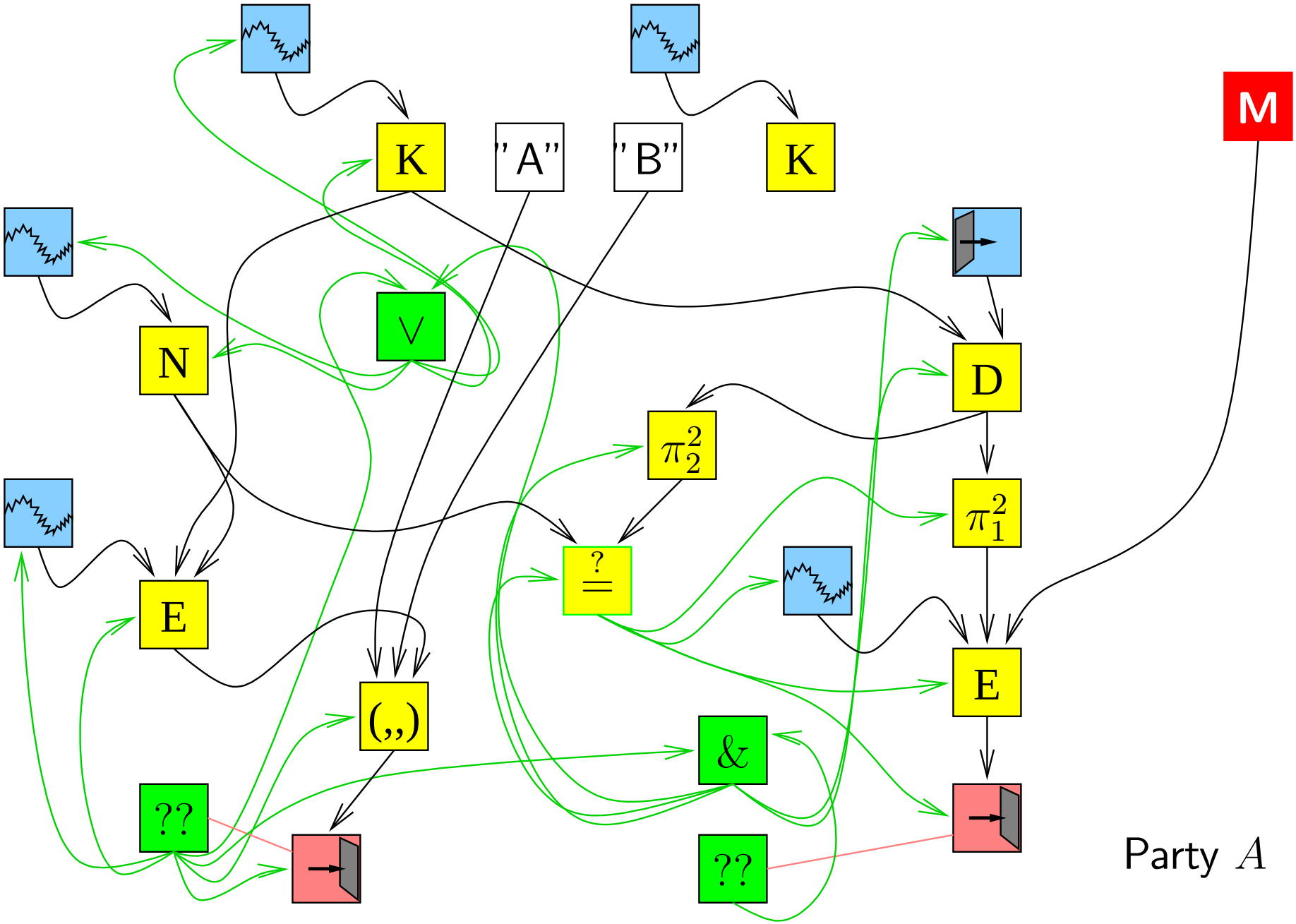


Generate keys K_{AS} and K_{BS}

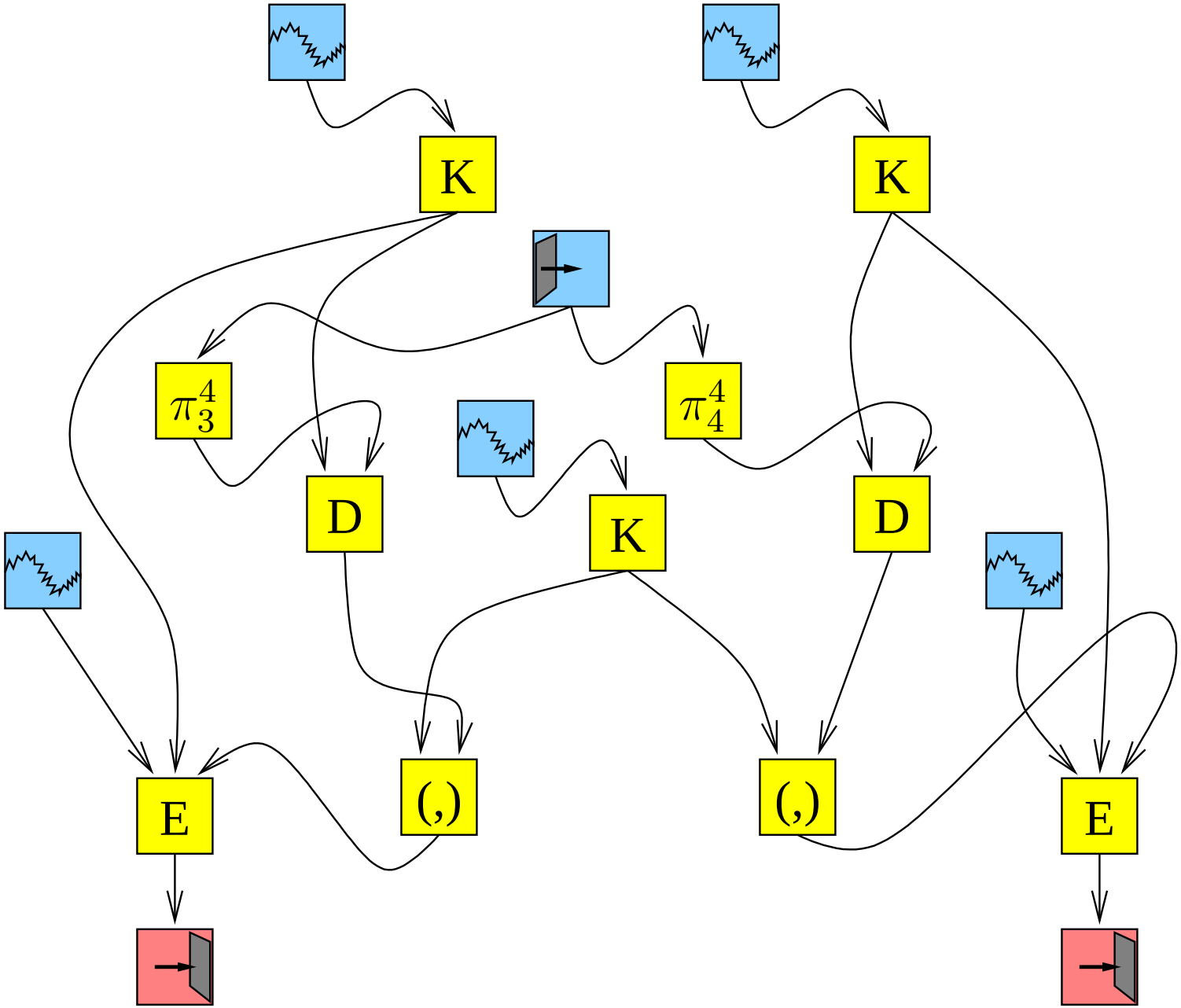
A protocol



A protocol

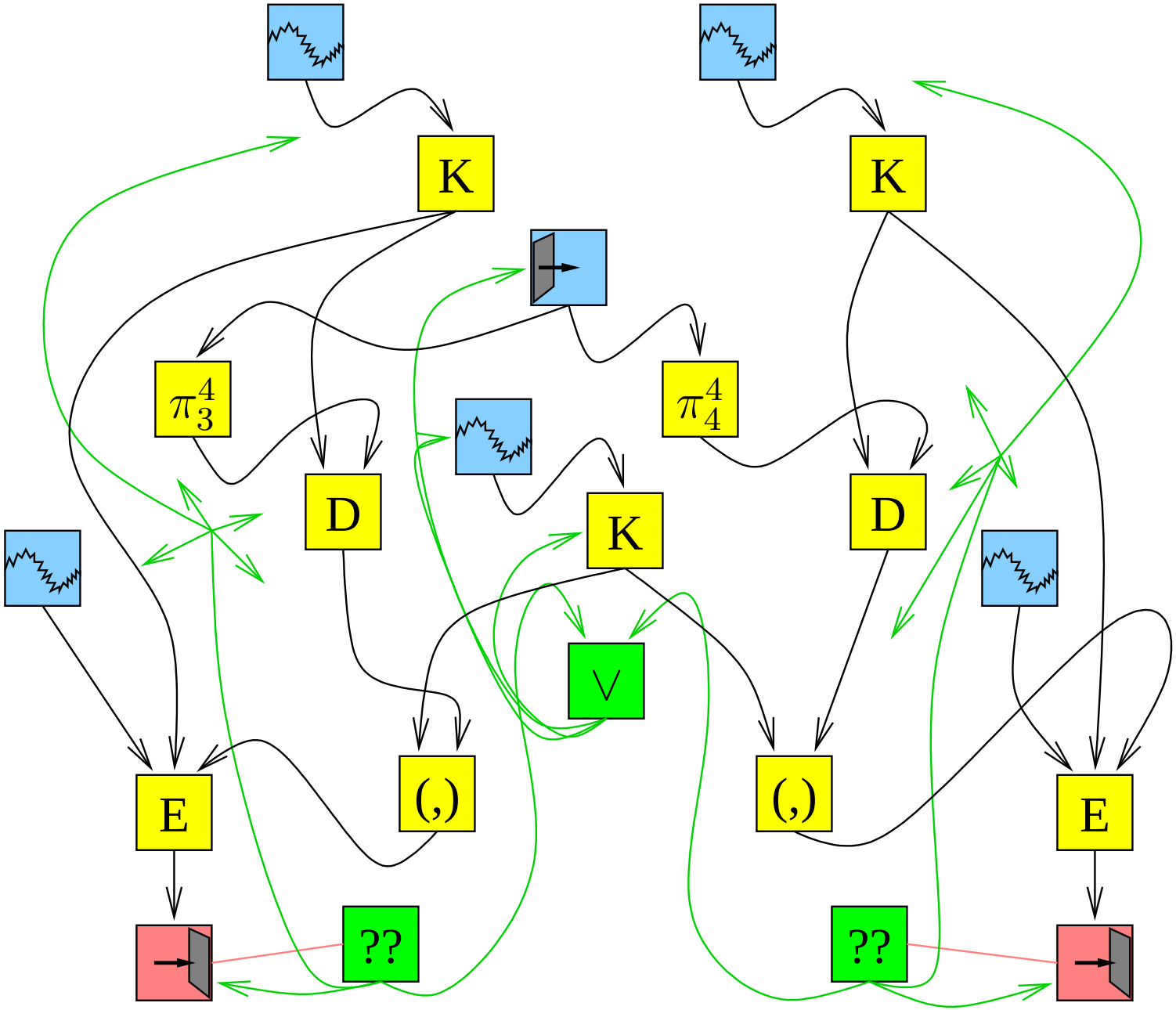


A protocol



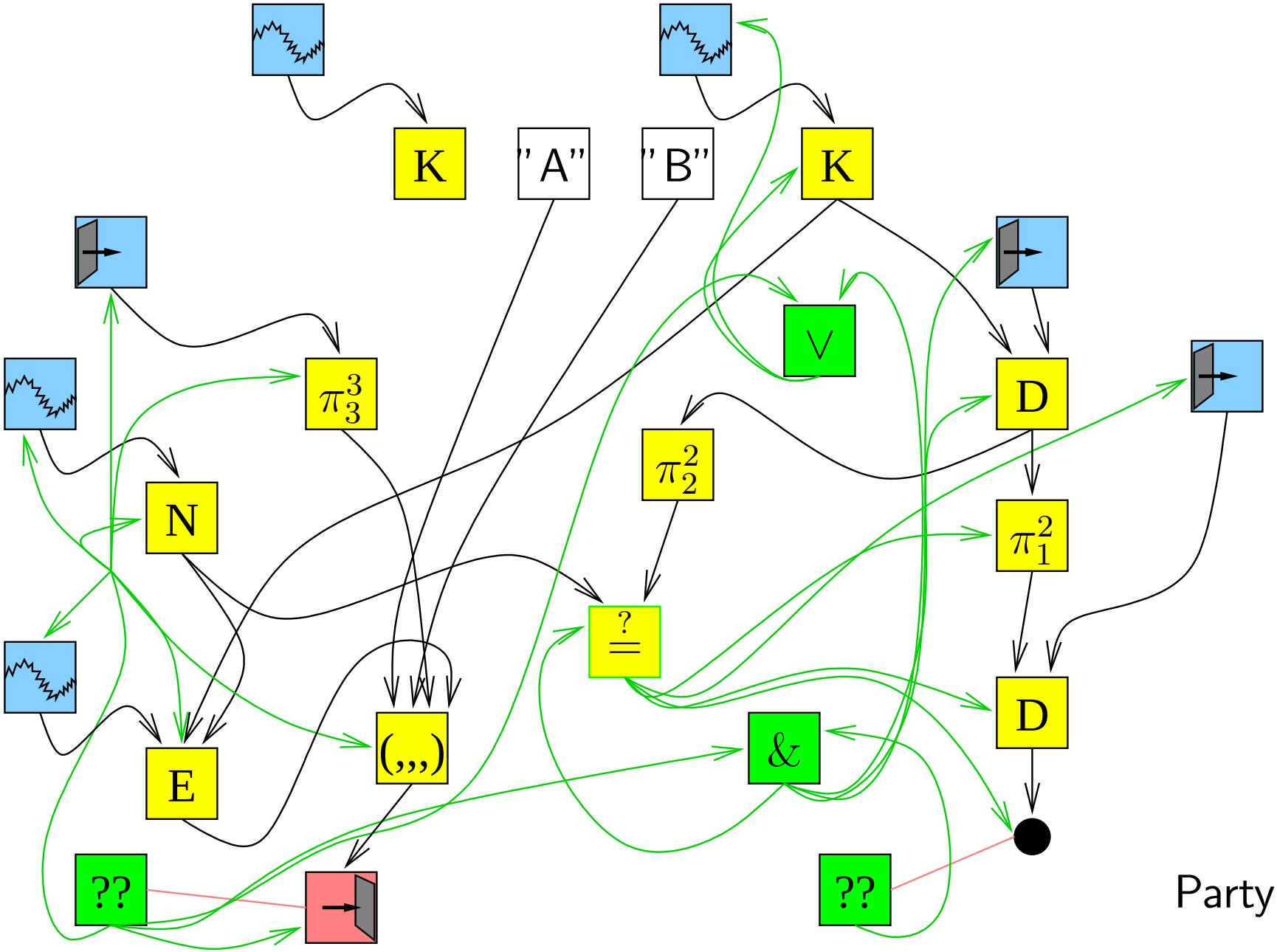
Party S

A protocol

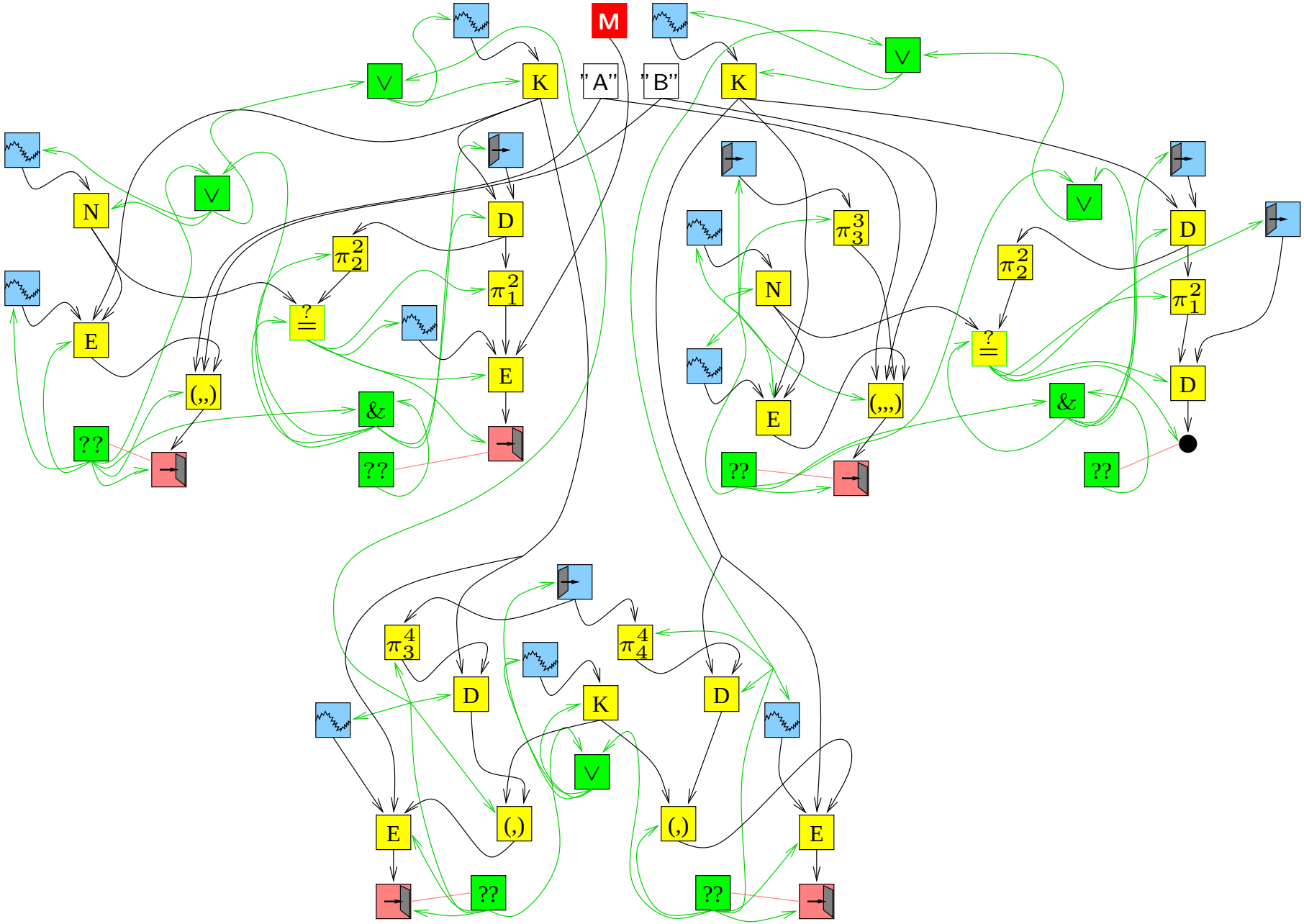


Party S

A protocol



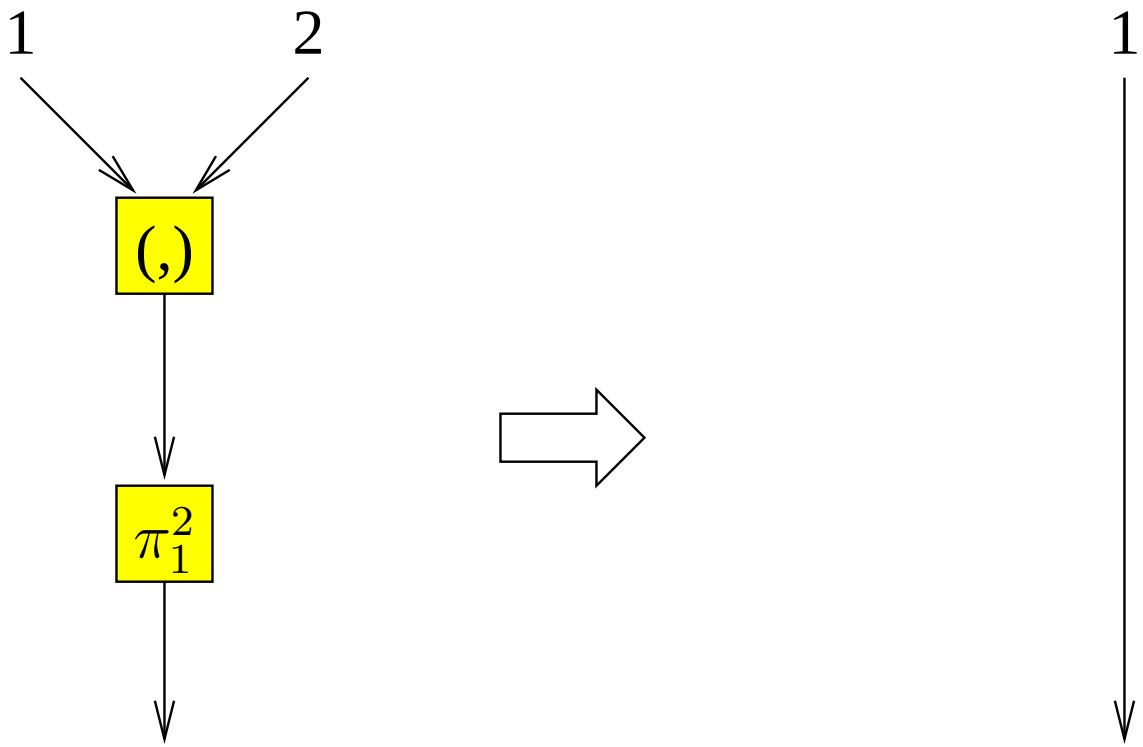
A protocol



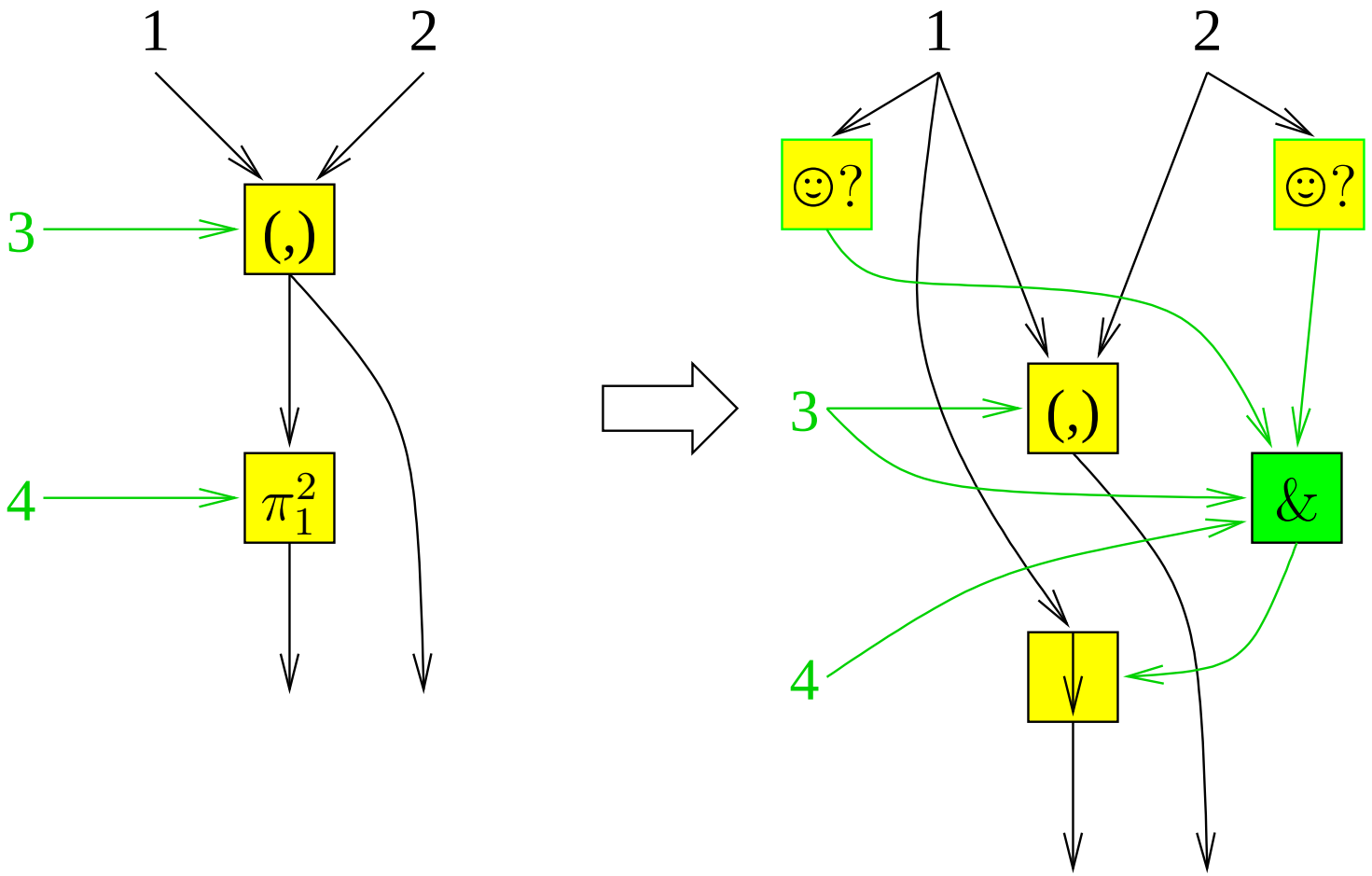
Good sides

- The structure of definitions and uses of values is explicit.
  - No copying of values.
  - No variable names at all...
- We immediately see what is used where.
  - ...which greatly simplifies finding out whether some cryptographic reduction is allowed.
  - ...and also helps doing other simplifications.

Some obvious simplifications

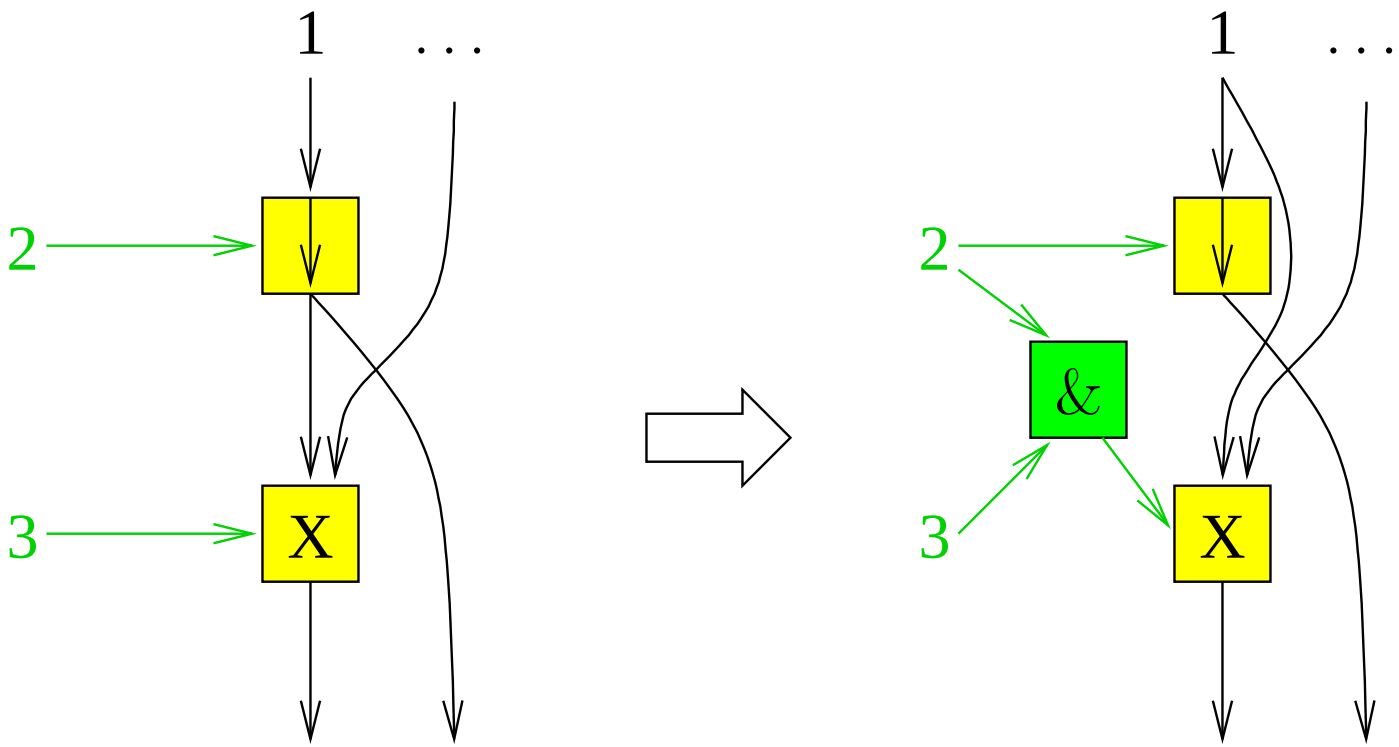


Some obvious simplifications



We can do dead code elimination afterwards.

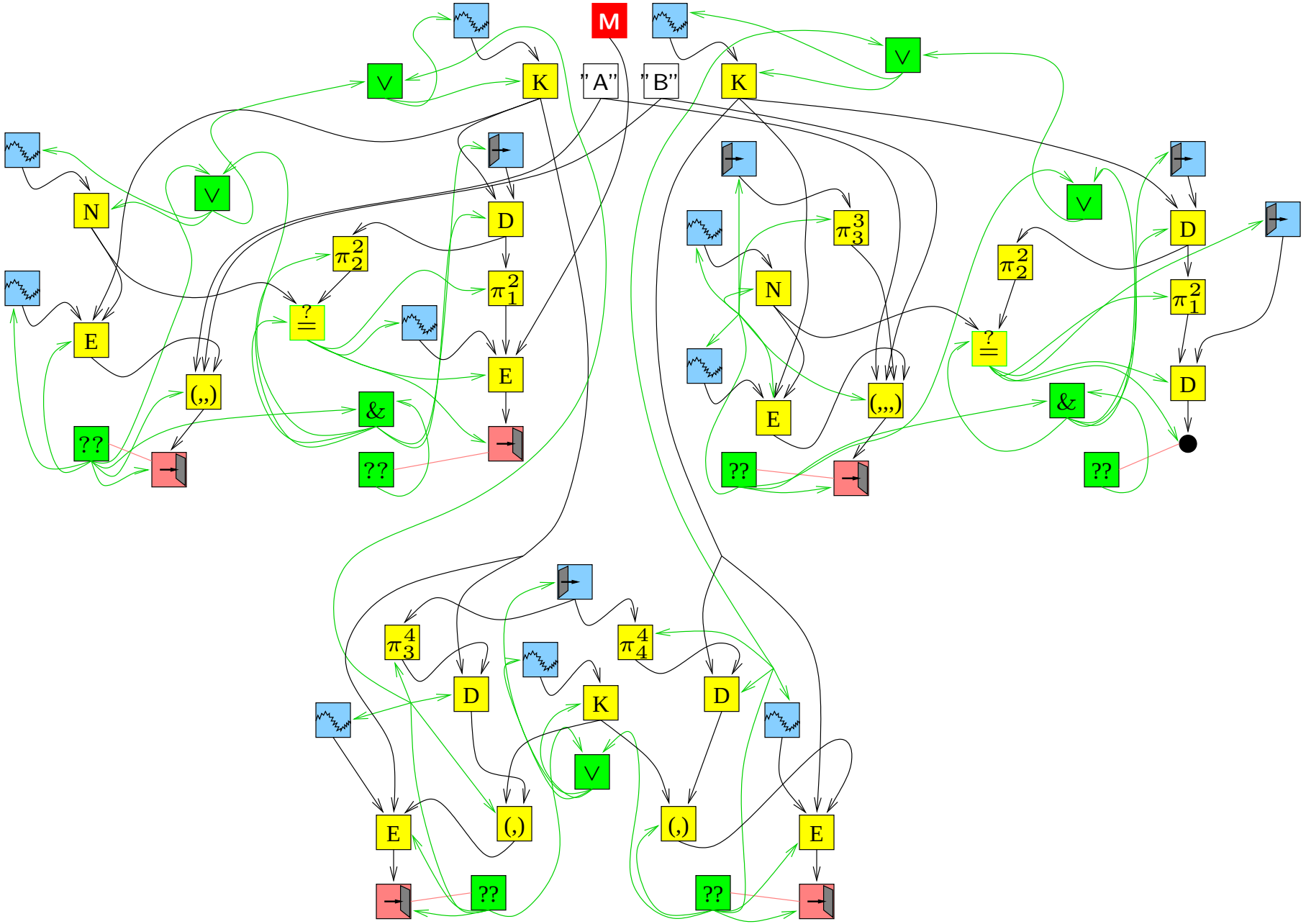
Some obvious simplifications



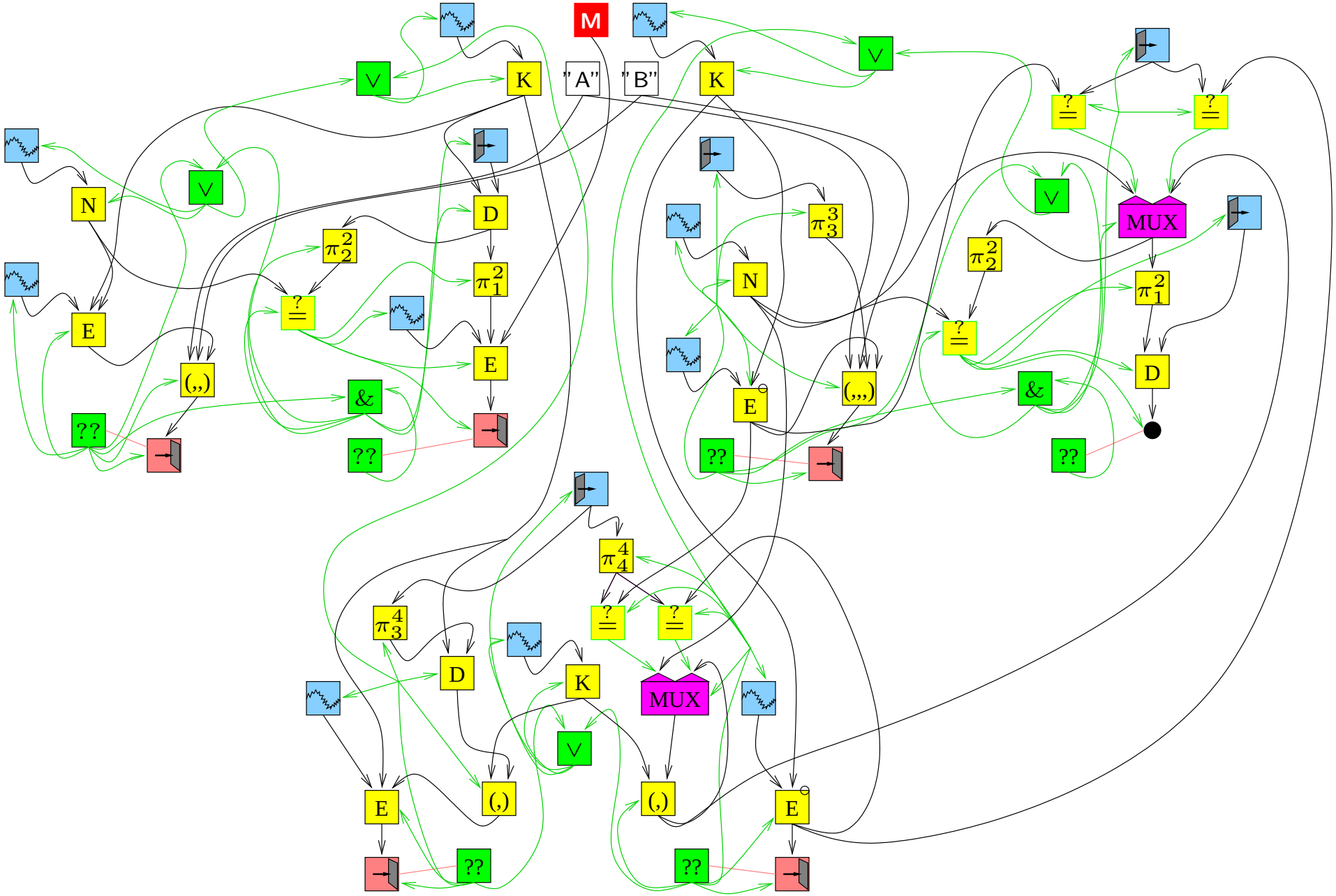
Simplifying encryption

- If the symmetric encryption is IND-CCA and INT-CTXT secure then we can replace the encryptions and decryptions as follows:
  - Encryptions — replace the plaintext with some constant $\mathbf{0}$.
  - Decryptions — replace them by
    - comparing the ciphertext with the results of all encryptions (with the same key);
    - if there is a match then take the corresponding (original) plaintext as the result;
    - if there is no match then fail.
- . . . provided that the key is used only for encrypting and decrypting.

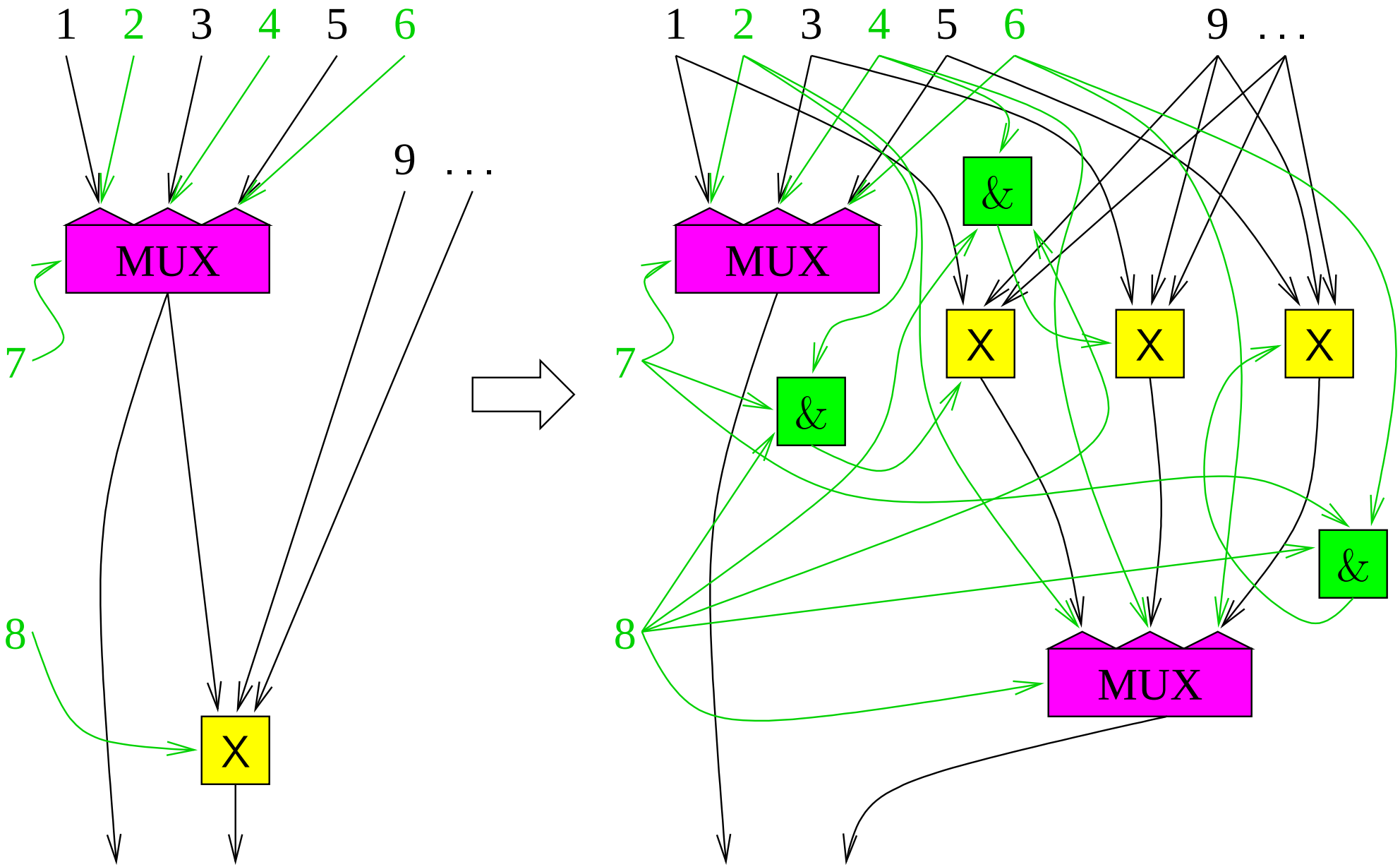
Which keys can be “replaced”



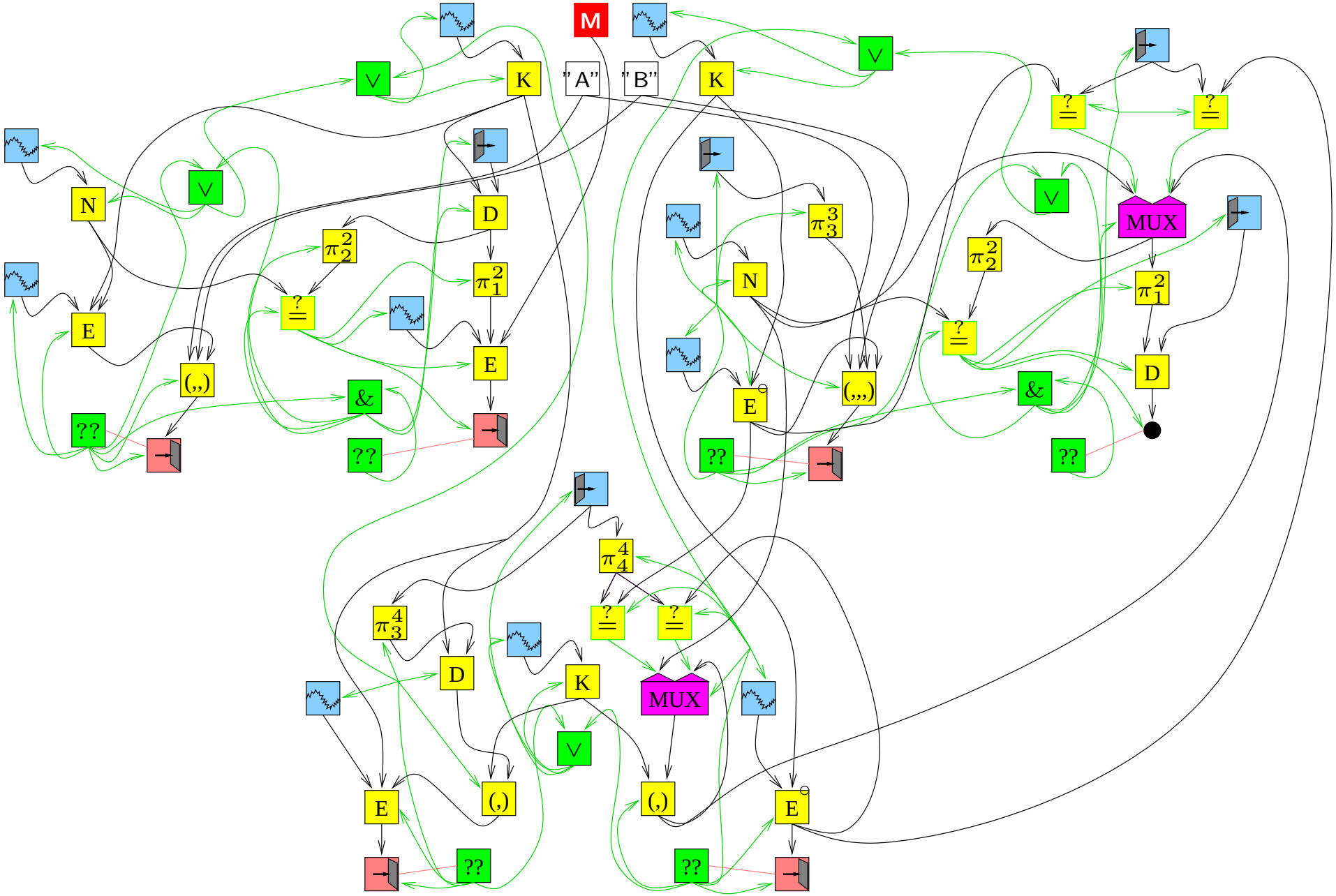
Replace K_{BS}


M
"A"
"B"
K
K
∨
∨
N
E
(,,)
??
&
??
D
π^2_2
π^2_1
?
=
E
π^3_3
N
E
(,,)
??
?
=
?
=
∨
MUX
π^2_2
π^2_1
D
&
??
π^4_4
π^4_3
?
=
?
=
D
K
MUX
∨
E
(,)
(,)
E
??
??

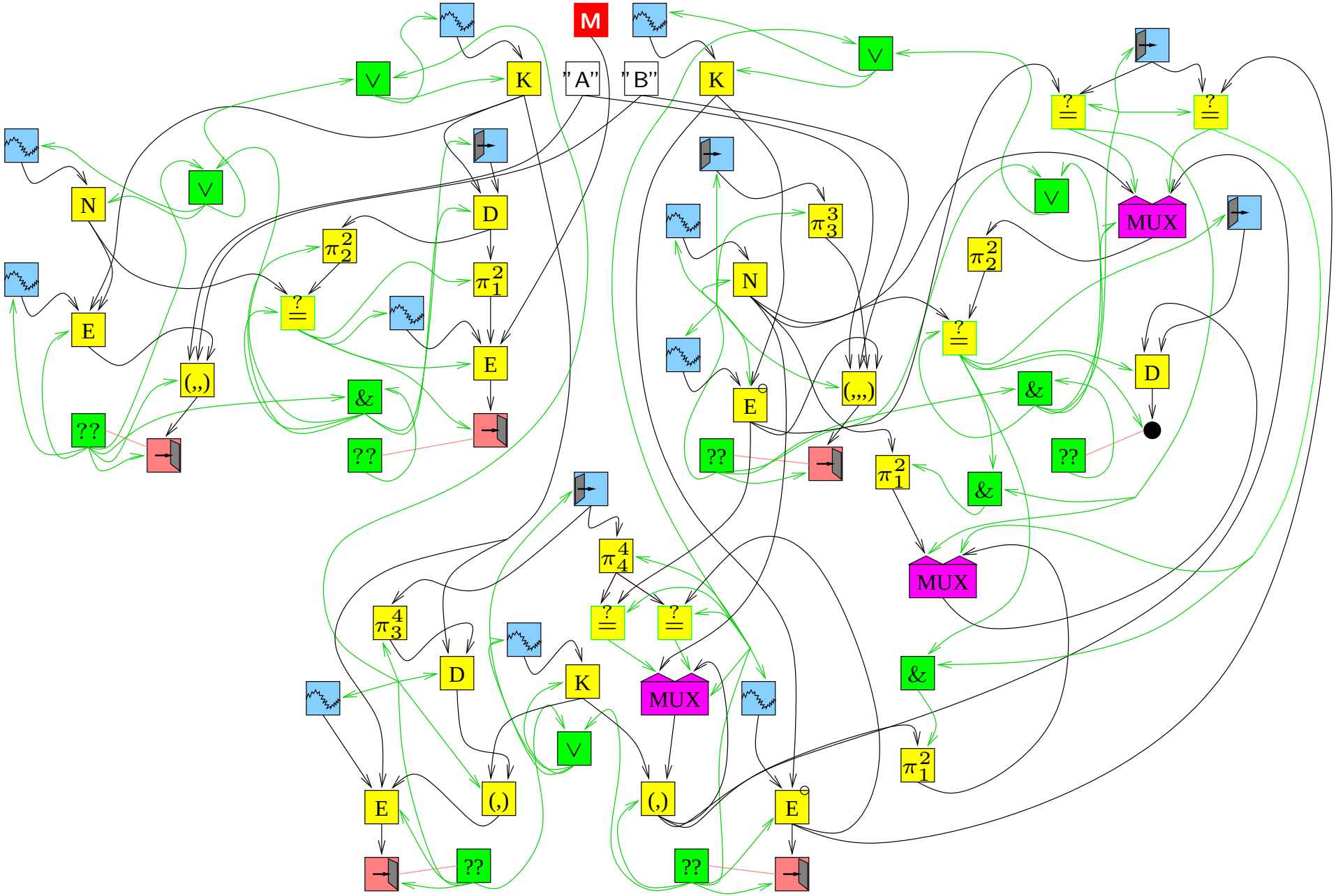
Computation $\leftrightarrow$ MUX



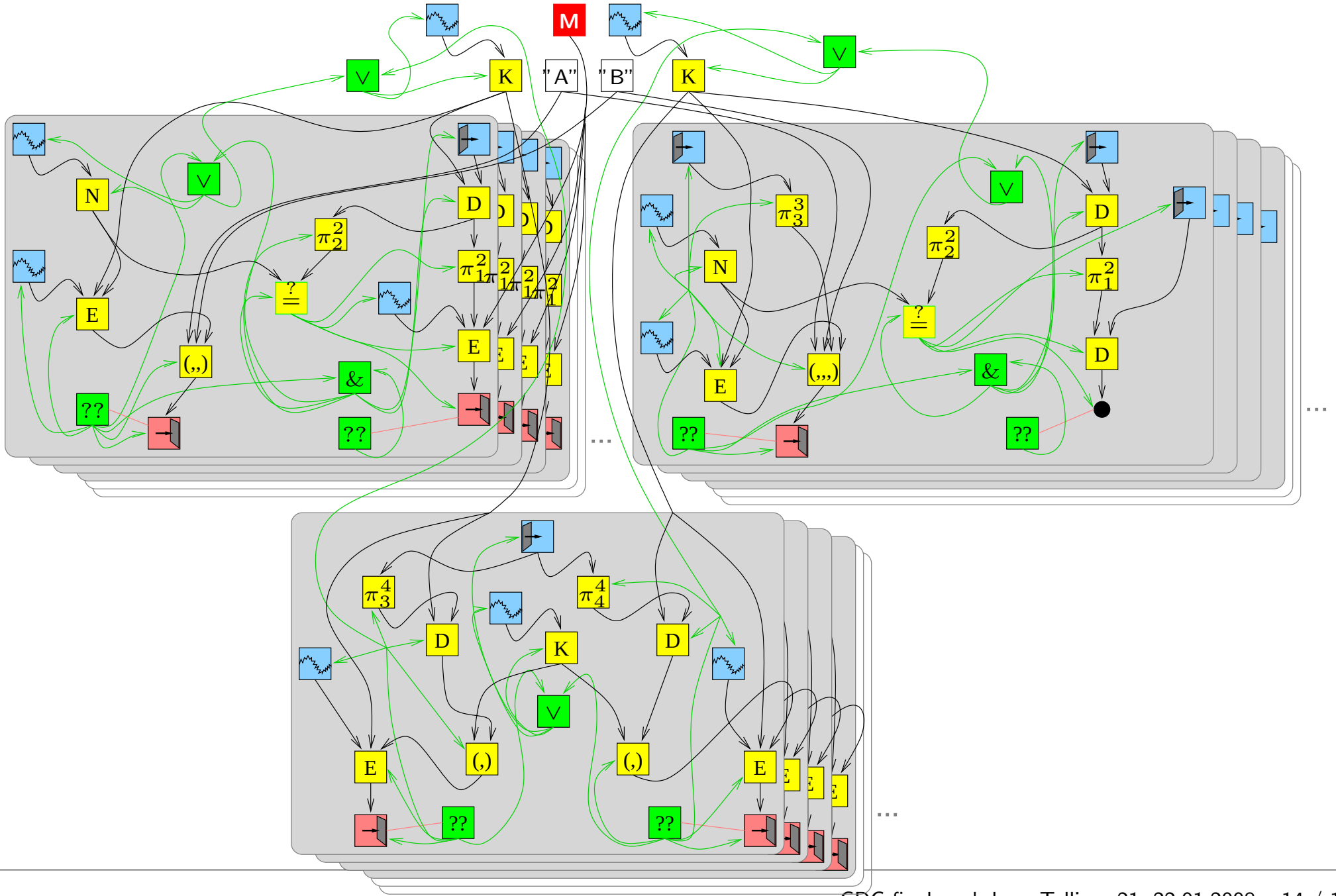
Application. . .



Application. . .



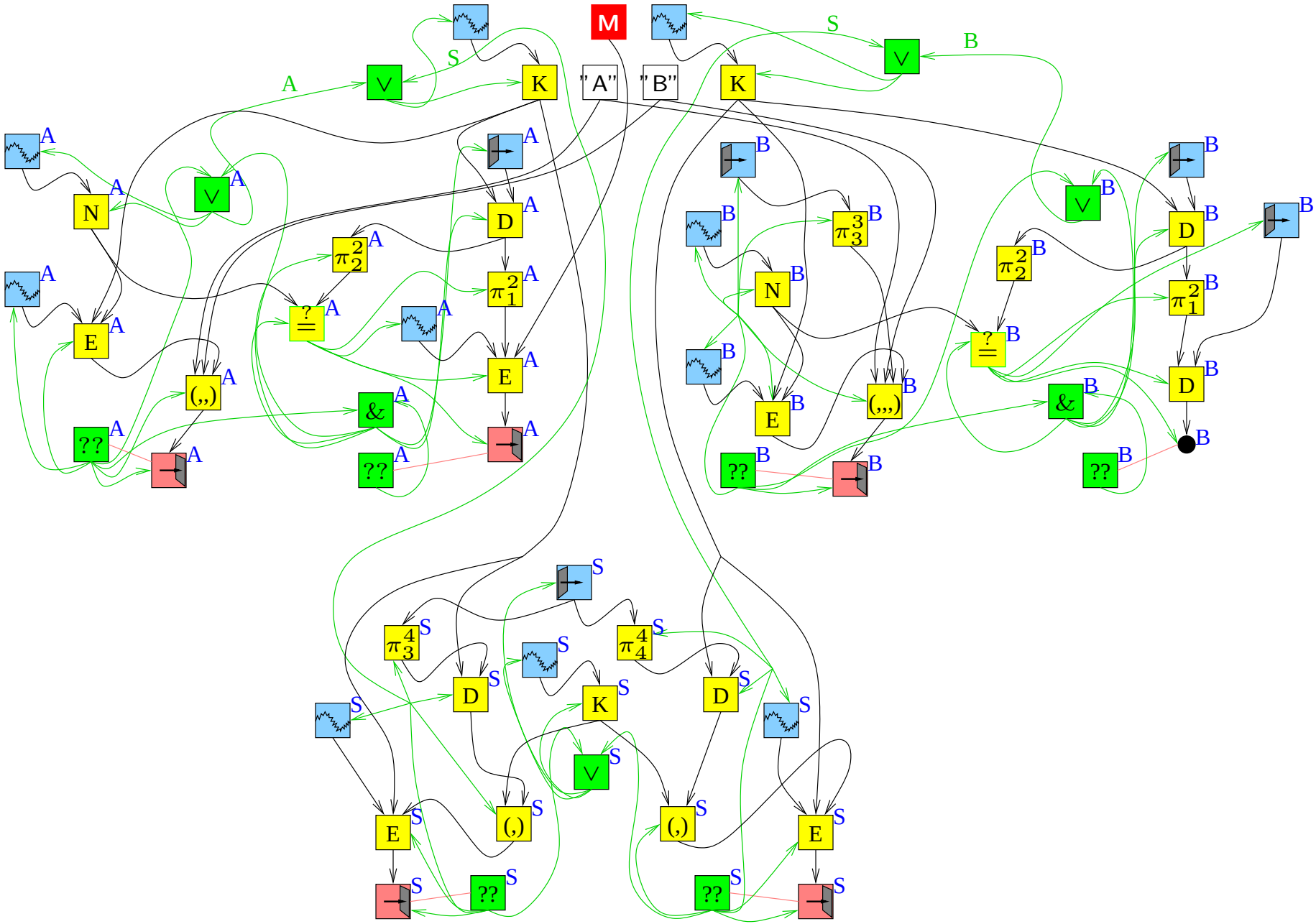
Replication



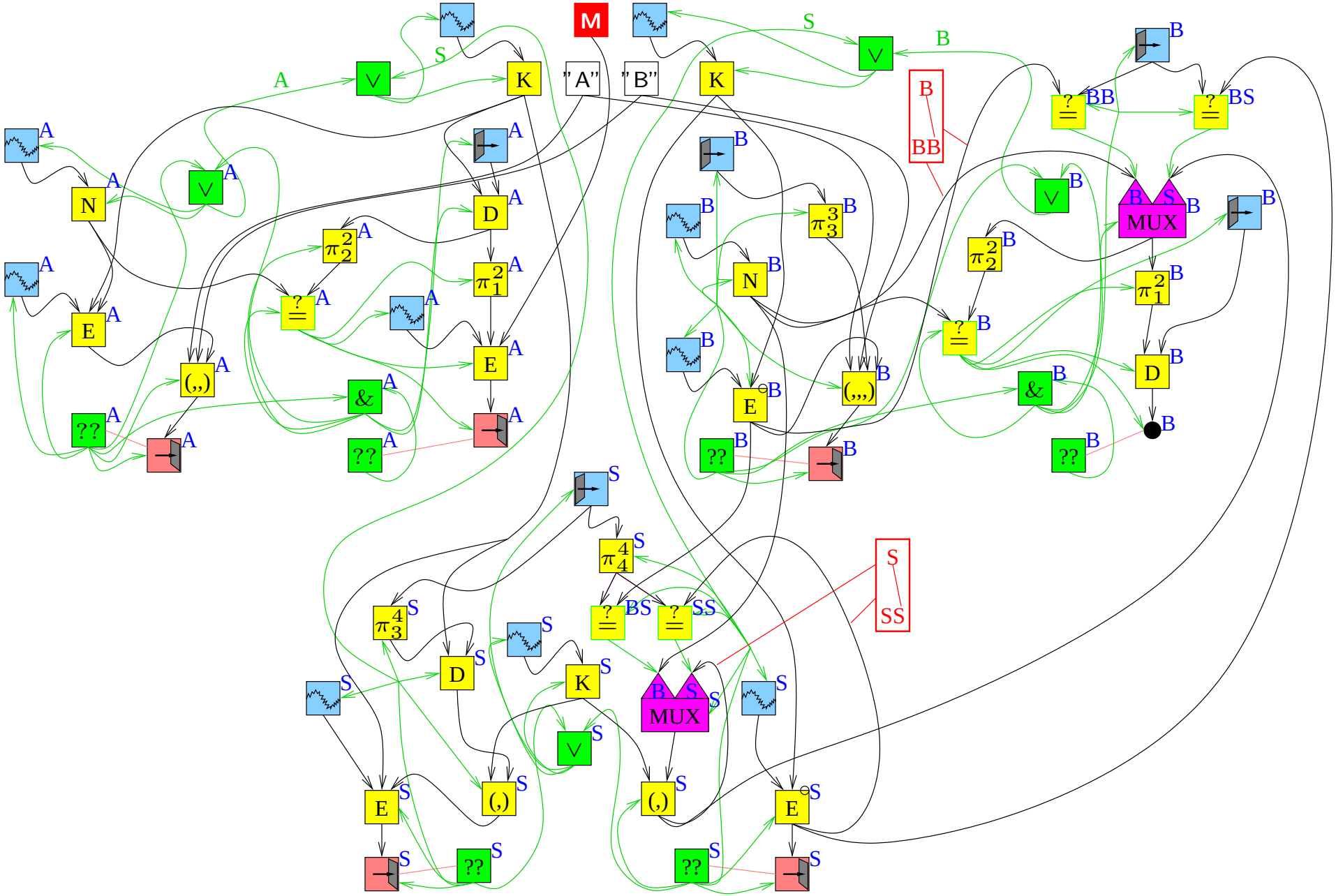
Representing infinite graphs

- Nodes in different planes, but in the same position are represented by a single node.
  - Such nodes are *one-dimensional*.
- There may be replication inside replication.
  - The corresponding nodes in the representation have more than one dimension.
- In the representation, the edges are equipped with *coordinate mappings*.
- In the representation, the edges generally cannot go from a higher-dimensional node to lower-dimensional node.
  - Exceptions: target node is an *infinite or* or MUX.
  - Then we record which dimensions are contracted.

Example



After replacing K_{BS}



And so it goes...

- The analyzer has been implemented.
  - about 9000 lines of OCaml
- We can analyse protocols for secrecy, as well as integrity (correspondence) properties.
- The previous example takes less than a minute on this laptop.
- A typical public-key protocol (Needham-Schroeder-Lowe) takes about 6 minutes.
- Some global analyses are necessary, these take the most time.
  - For which (v_1, v_2) does "v_1 true $\Rightarrow$ v_2 true" hold?
  - For which (v_1, v_2) does "v_1 true $\Rightarrow$ v_2 false" hold?
- The implementation of these analyses is currently not optimal.
  - A good implementation would reduce the running times reported above.