Reliability

In case of a crash, recover to a consistent (or correct state) and continue processing.

Types of Failures

1. Node failure
2. Communication line of failure
3. Loss of a message (or transaction)
4. Network partition
5. Any combination of above

Approaches to Reliability

1. Audit trails (or logs)
2. Two phase commit protocol
3. Retry based on timing mechanism
4. Reconfigure
5. Allow enough concurrency which permits definite recovery (avoid certain types of conflicting parallelism)
6. Crash resistance design

Recovery Controller

Types of failures:

- transaction failure
- site failure (local or remote)
- communication system failure

Transaction failure

UNDO/REDO Logs (Gray)

transparent transaction

(effects of execution in private workspace)

⇒ Failure does not affect the rest of the system

Site failure

volatile storage lost

stable storage lost

processing capability lost

(no new transactions accepted)

System Restart

Types of transactions:

1. In commitment phase
2. Committed actions reflected in real/stable
3. Have not yet begun
4. In prelude (have done only undoable actions)

We need:

stable undo log;

stable redo log (at commit);

perform redo log (after commit)

Problem:

entry into undo log; performing the action

Solution:

undo actions $\neg < T, A, E >$

must be restartable (or idempotent)

DO – UNDO

≡ UNDO

≡ DO – UNDO – UNDO – UNDO --- UNDO

Local site failure

- Transaction committed $\Rightarrow$ do nothing
- Transaction semi-committed $\Rightarrow$ abort
- Transaction computing/validating $\Rightarrow$ abort

AVOIDS BLOCKING

Remote site failure

- Assume failed site will accept transaction
- Send abort/commit messages to failed site via spoolers

Initialization of failed site

- Update for globally committed transaction before validating other transactions
- If spooler crashed, request other sites to send list of committed transactions

Communication system failure

- Network partition
- Lost message
- Message order messed up

Network partition

- Semi-commit in all partitions and commit on reconnection (updates available to user with warning)
- Commit transactions if primary copy taken for all entities within the partition
- Consider commutative actions
- Compensating transactions

Compensating transactions

- Commit transactions in all partitions
- Break cycle by removing semi-committed transactions
- Otherwise abort transactions that are invisible to the environment

  (no incident edges)
- Pay the price of committing such transactions and issue compensating transactions

Recomputing cost

- Size of readset/writeset
- Computation complexity

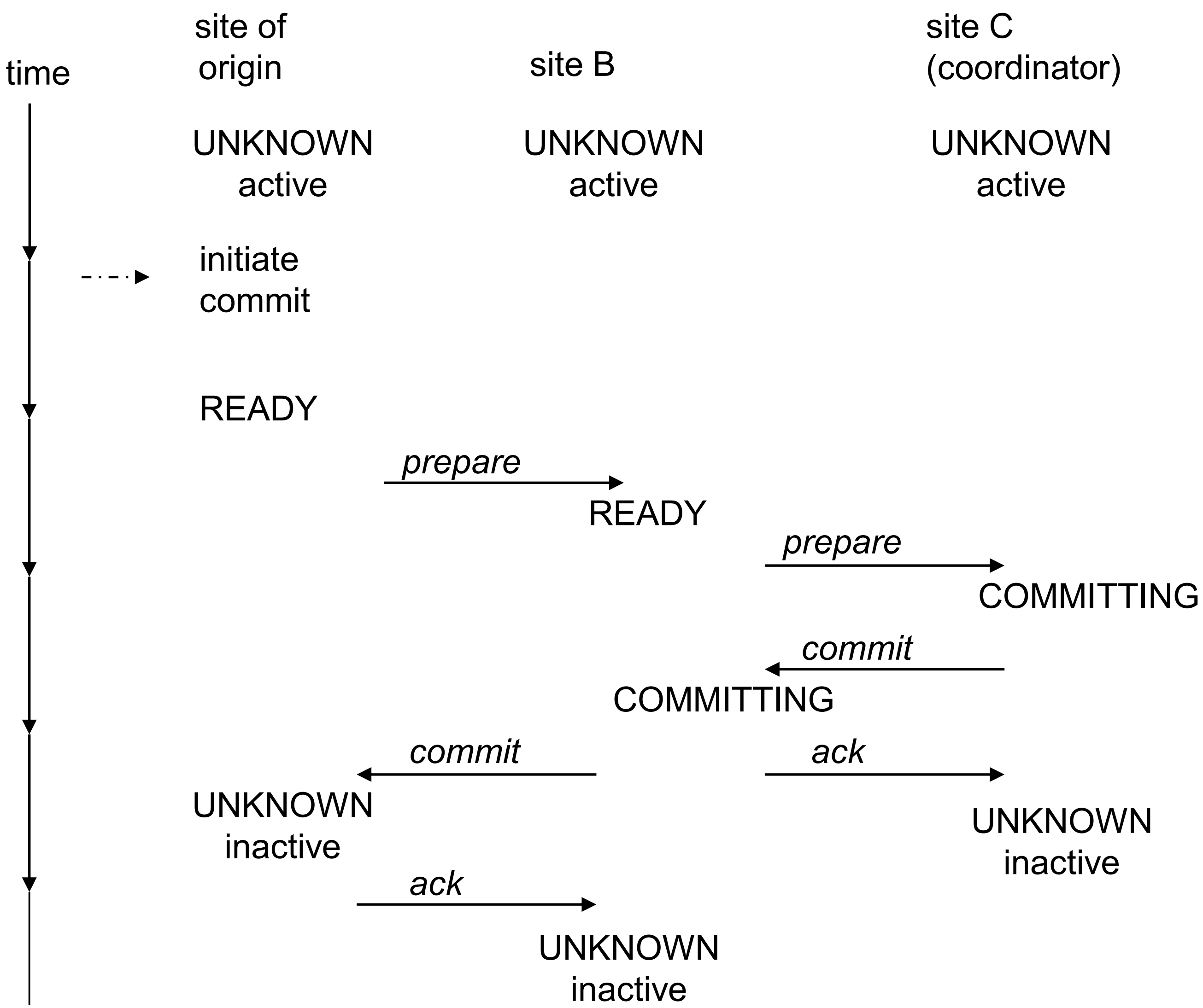


Figure 5.3: Linear Commit Protocol

TABLE 1: Local Site Failure

Local Site Failure	System's Decision at Local Site
After Committing/Aborting a local transaction	Do nothing (Assume: Message has been sent to remote sites)
After Semi-Committing a local transaction	Abort transaction when local site recovers Send abort messages to other sites
During computing/validating a local transaction	Abort transaction when local site recovers Send abort message to other sites

- Ripple Edges:

  T_i reads a value produced by T_j in same partition

- Precedence Edges:

  Ti reads a value but has now been changed by Tj in same partition

- Interference Edges:

  T_i reads a data-item in one partition and T_j writes in another partition then $T_i \rightarrow T_j$

Finding minimal number of nodes to break all cycles in a precedence graph consisting of only two-cycle of ripple edges has a polynomial solver.

- Communications
  - Design
    - Sockets, ports, calls (sendto, recvfrom)
    - Oracle
    - Server cache
    - Addressing in RAID
    - LUDP
  - High level calls
    - Setup
    - RegisterSelf
    - ServActive
    - ServAddr
    - SendPacket
    - RecvMsg
- Software guide (where is the code and how is it compiled?)
- Testing RAID
  - RAID installation
  - RAIDTOol
  - Example test session
- Recommended reading
- How to incorporate a new server (RC)
- How to run an experiment (John-Comm)

- Storage of backup copies of database
  - Reduce storage
  - Maintain number of versions
  - Access time
- Move servers at Kernel level
  - Buffer pool, scheduler, lightweight processes
  - Shared memory

- New protocols and algorithms
  Replicated copy control

  - Survivability
  - Availability
  - Reconfigurability
  - Consistency and dependability
  - Performance

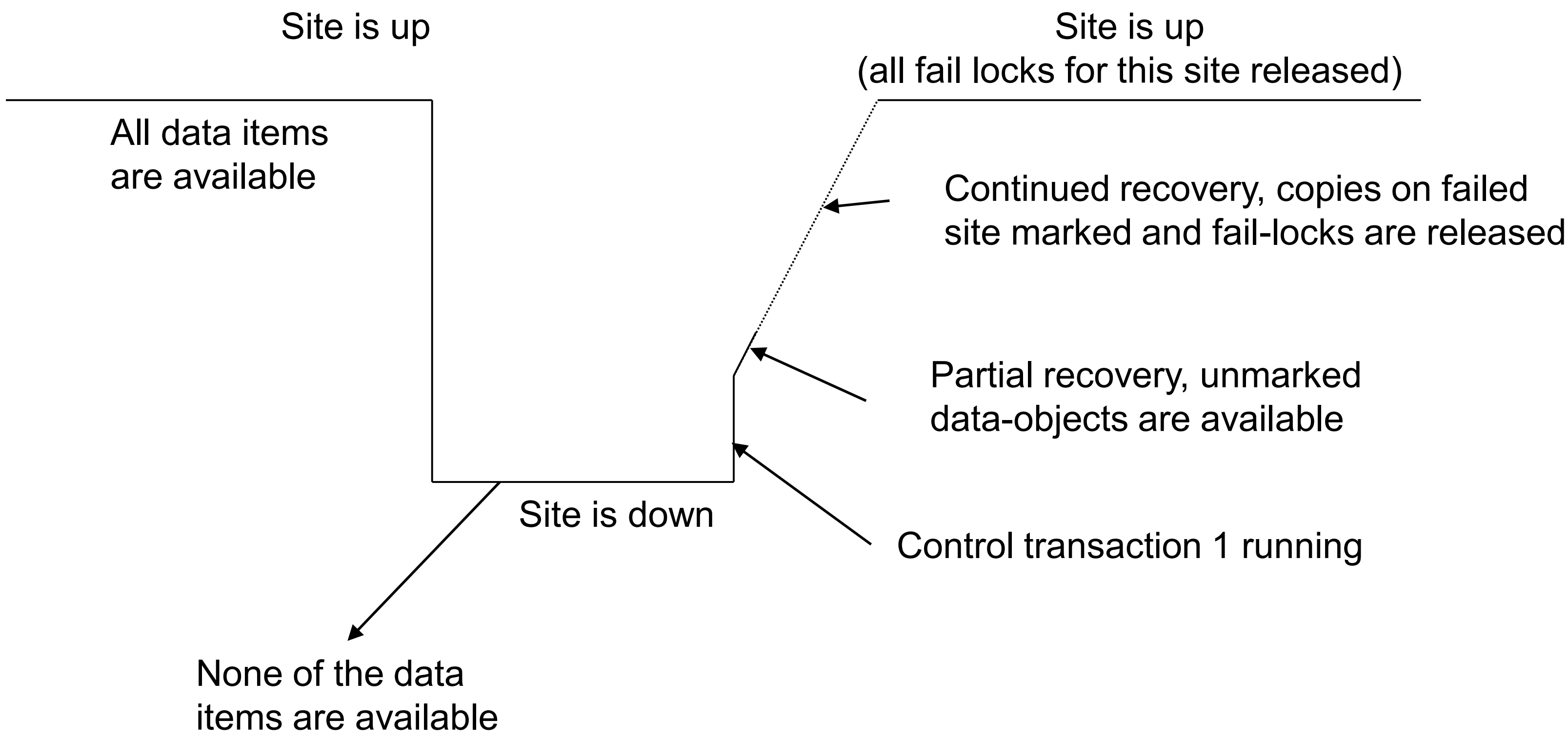
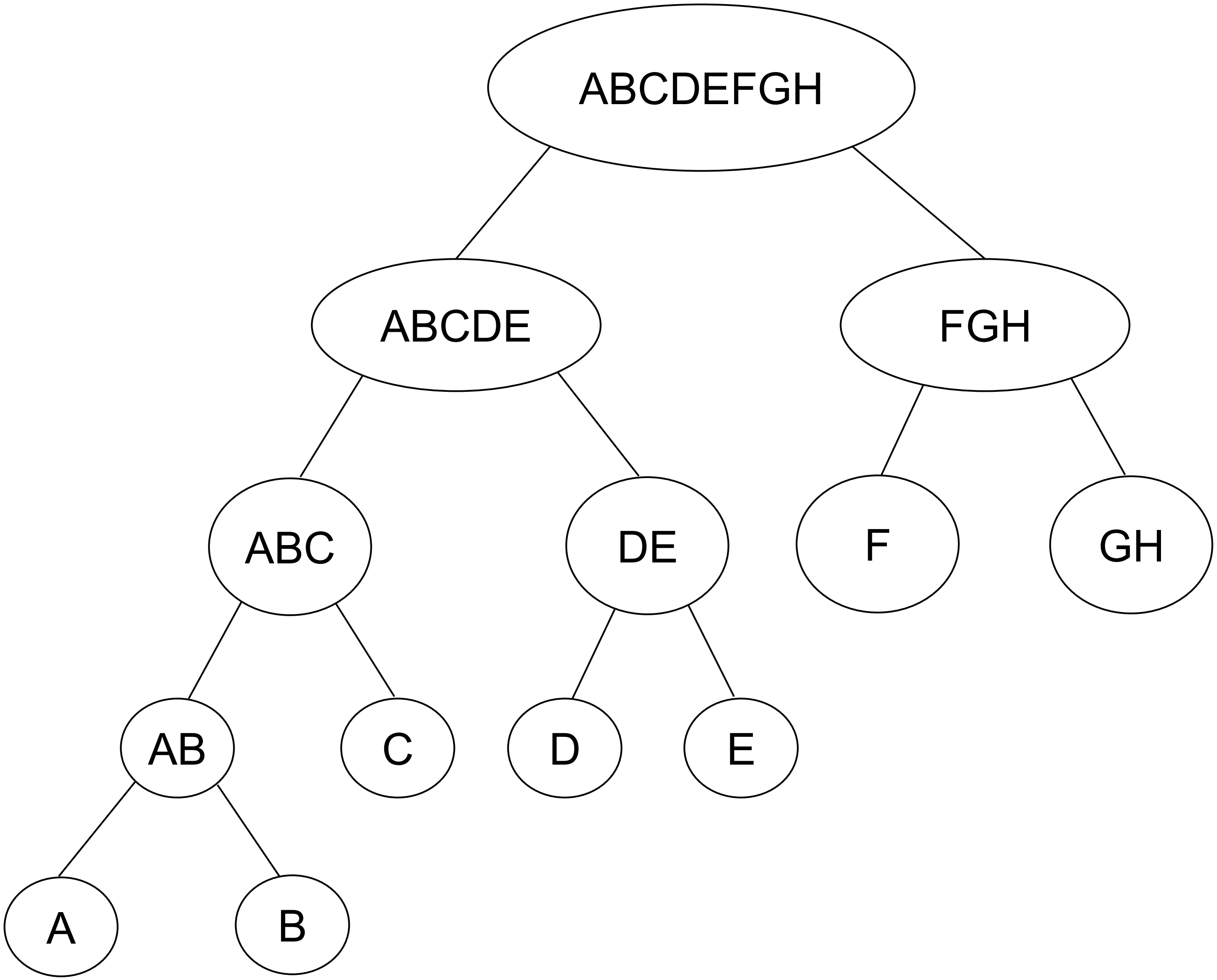
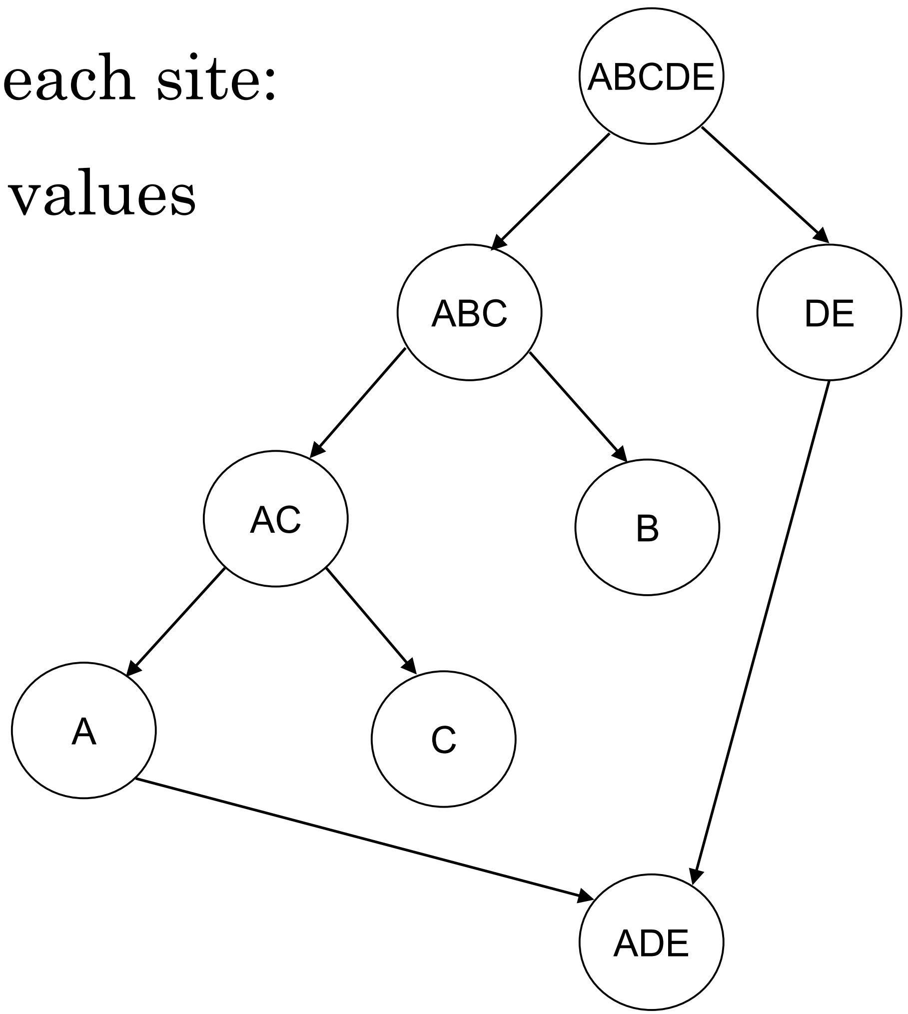


Figure : States in site recovery and availability of data-items for transaction processing


ABCDEFGH
ABCDE
FGH
ABC
DE
F
GH
AB
C
D
E
A
B

Data Structures

- Connection vector at each site:
  - Vector of boolean values
- Partition graph



- Site name vector of file f

  (n is the number of copies)

$$S = < s_1, s_2 ,\ldots, s_n >$$

- Linear order vector of file f

$$L = < l_1, l_2 ,\ldots, l_n >$$

- Version number X of a copy of file f

  Number of times network partitioned while the copy is in majority

- Version vector of a copy at site S_i

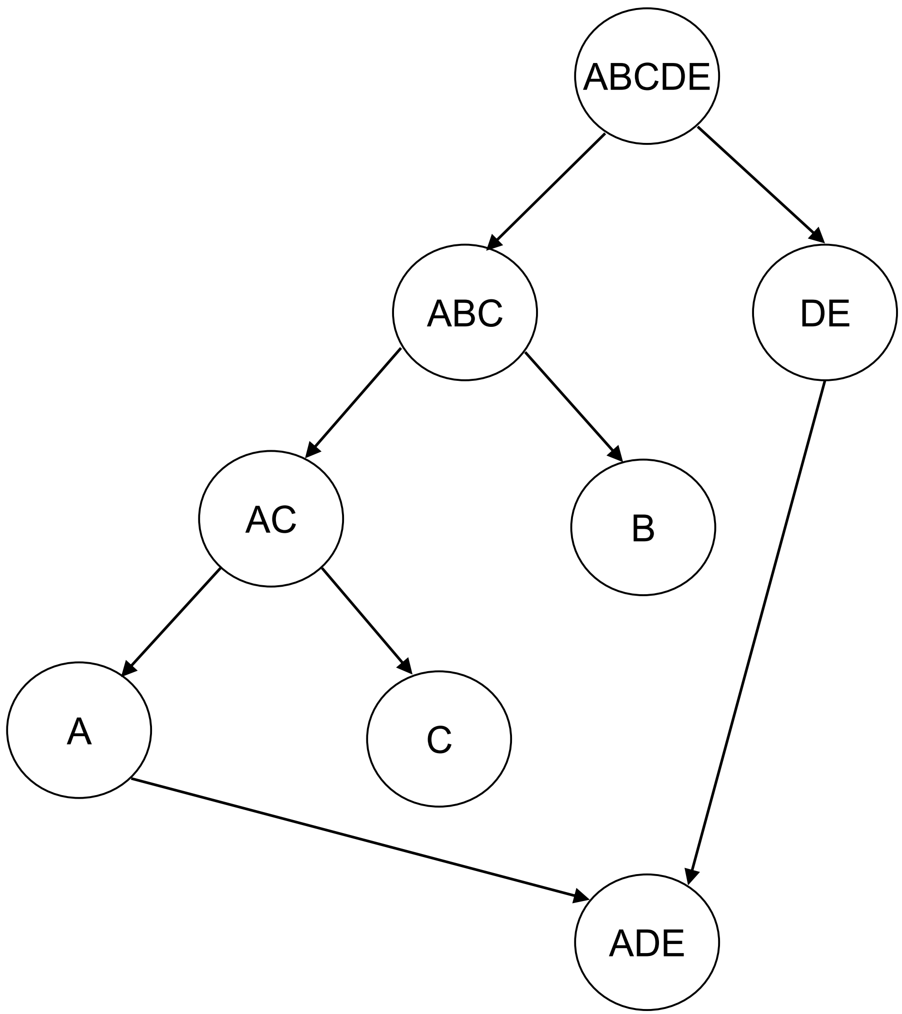
$$V = < v_1, v_2 ,\ldots, v_n >$$

- Marked vector of a copy of file f

$$M = < M_1, m_2 ,\ldots, m_n >$$

$$m_i \quad = T \text{ if } \textit{marked}$$

$$\quad\quad = F \text{ if } \textit{unmarked}$$


ABCDE
ABC
DE
AC
B
A
C
ADE

Examples of Partition Trees

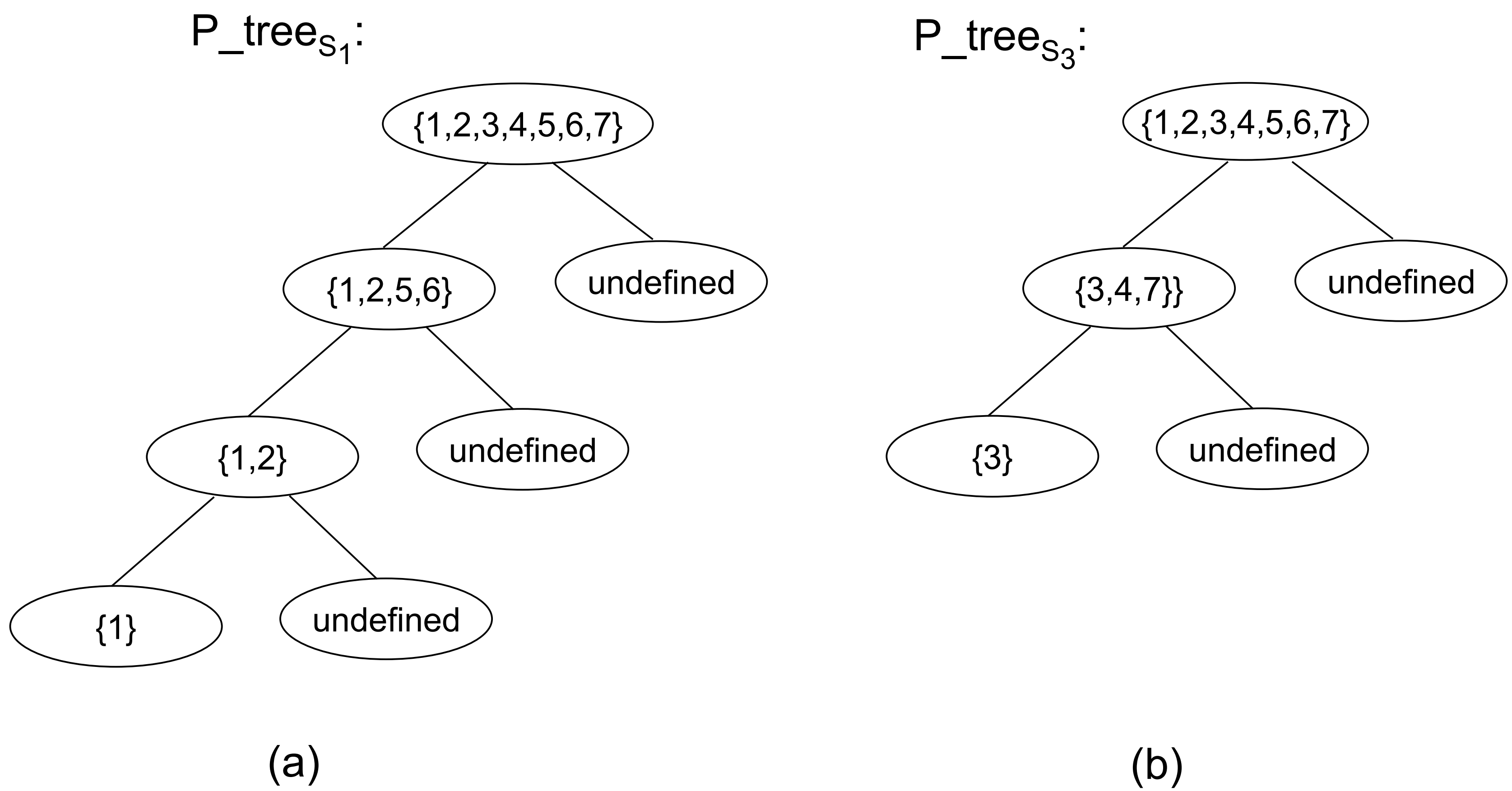


Figure 9. Partition trees maintained at S_1 and S_3 before any merge of partition occurs

Partition Tree after Merge

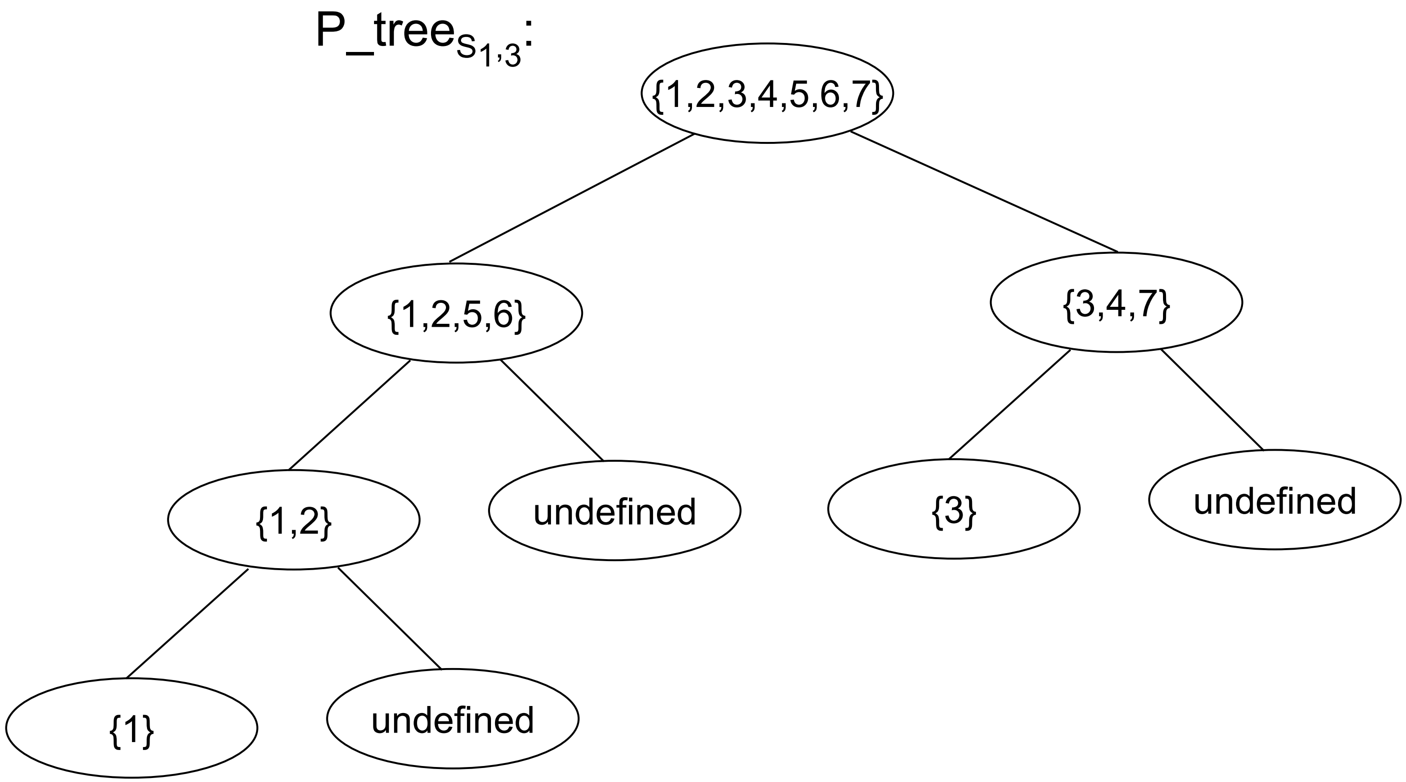


Figure 10. Partition tree maintained at S_1 and/or S_3 after S_3 merge