Checkpoint e Dump

- Il log serve a "**ricostruire**" le **operazioni**
- **Checkpoint** e **dump** servono ad evitare che la ricostruzione debba partire dall'inizio dei tempi
  - si usano con riferimento a tipi di guasti diversi
- L'**operazione di checkpoint** serve a "fare il punto" della situazione, semplificando le successive operazioni di ripristino
  - Ha lo scopo di **registrare** quali **transazioni** sono **attive** in un **certo istante**, cioè le transazioni "a metà strada"
  - Ha lo scopo duale di **confermare** che le **altre** o **non sono iniziate** o sono **finite**
    - Per tutte le transazioni che hanno **effettuato** il **commit** i **dati** possono essere **trasferiti** in **memoria di massa**

Operazione di checkpoint

- **Varie modalità**, vediamo la più semplice:
  - Si **sospende l'accettazione** delle operazioni di **commit** o **abort** da parte delle transazioni
  - Si **forza** la **scrittura** in **memoria di massa** delle pagine in memoria modificate da **transazioni** che hanno già fatto **commit**
  - Si **forza** la **scrittura** nel **log** di un record contenente gli identificatori delle **transazioni attive**
  - Si **riprendono ad accettare** tutte le operazioni da parte delle transazioni
- Con questo funzionamento si **garantisce la persistenza** delle transazioni che hanno eseguito il commit

Dump

- **Copia completa** (“di riserva”, detta anche *backup*) della base di dati
  - Solitamente prodotta mentre il **sistema non è operativo**
  - Salvato in **memoria stabile**
  - Un **record di dump** nel log indica il momento in cui il dump è stato effettuato
    - e dettagli pratici, file, dispositivo, ...

Esito di una transazione

- **L'esito** di una transazione è **determinato irrevocabilmente** quando viene **scritto il record** di **commit** nel log in modo **sincrono**, con una *force*
  - una **guasto prima** di tale istante **porta ad un UNDO** di tutte le azioni, per ricostruire lo stato originario della base di dati
  - un **guasto successivo** non deve avere conseguenze: lo stato finale della base di dati deve essere ricostruito, con **REDO se necessario**
- I **record di abort** possono essere scritti in modo **asincrono**

Regole di modifica del log

- Regola **Write-Ahead-Log**:
  - si scrive la **parte BS** dei record del log prima di effettuare la corrispondente operazione sul database
  - consente di **disfare le azioni** già memorizzate (UNDO) di transazioni senza commit avendo in memoria stabile un valore corretto
- Regola **Commit-Precedenza**:
  - si scrive **la parte AS** dei record di log prima del commit
  - consente di **rifare le azioni** (REDO) di una transazione che ha effettuato il commit ma le cui pagine modificate non sono ancora state trascritte in memoria di massa

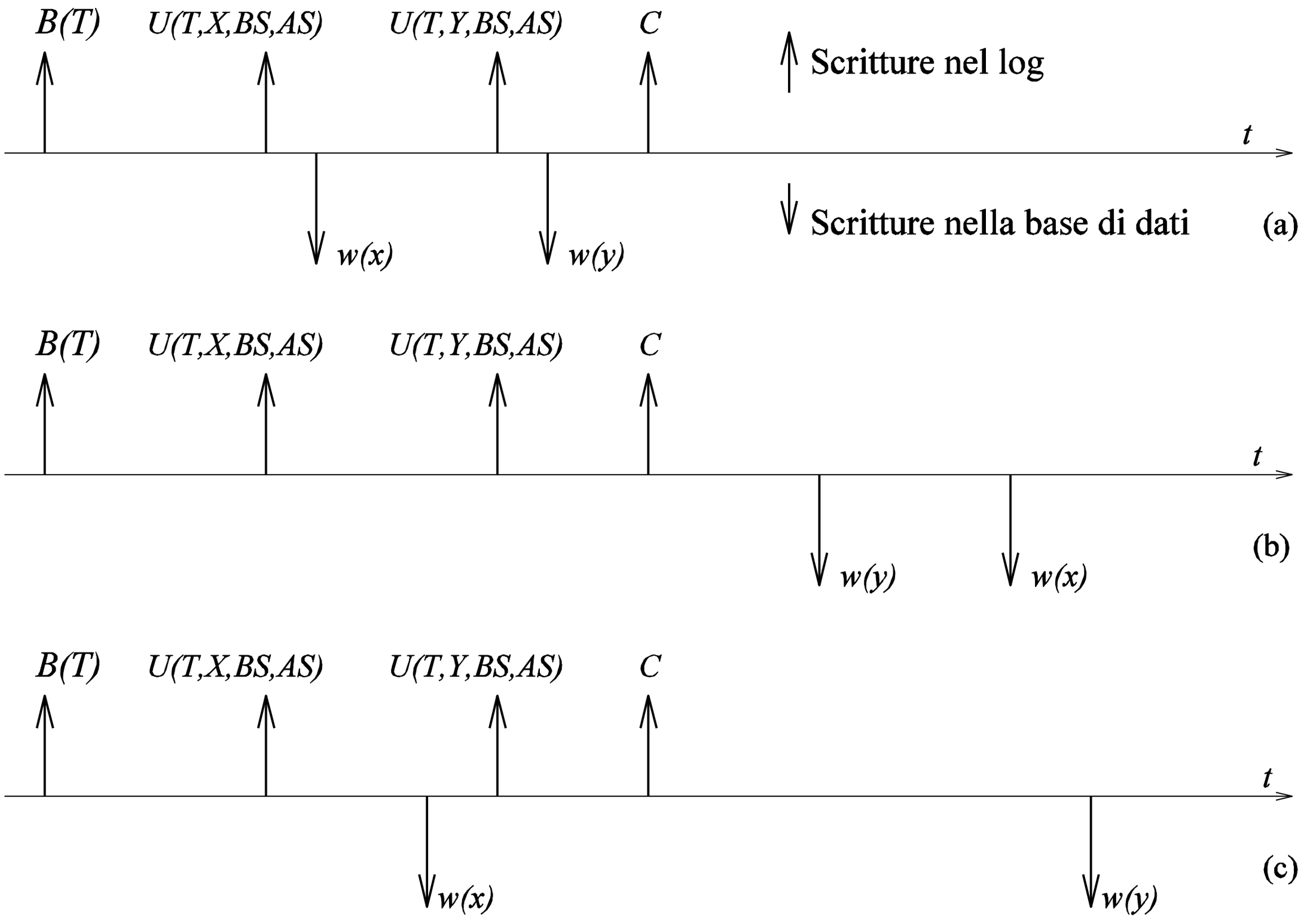
Operazioni UNDO e REDO

- **Undo** di una **azione** su un **oggetto** O:
  - `update`, `delete`: copiare il valore del before state (BS) nell'oggetto O
  - `insert`: eliminare O
- **Redo** di una **azione** su un **oggetto** O:
  - `insert`, `update`: copiare il valore dell'after state (AS) nell'oggetto O
  - `delete`: eliminare O
- **Idempotenza** di undo e redo:
  - $\text{undo}(\text{undo}(A)) = \text{undo}(A)$
  - $\text{redo}(\text{redo}(A)) = \text{redo}(A)$

E la base di dati?

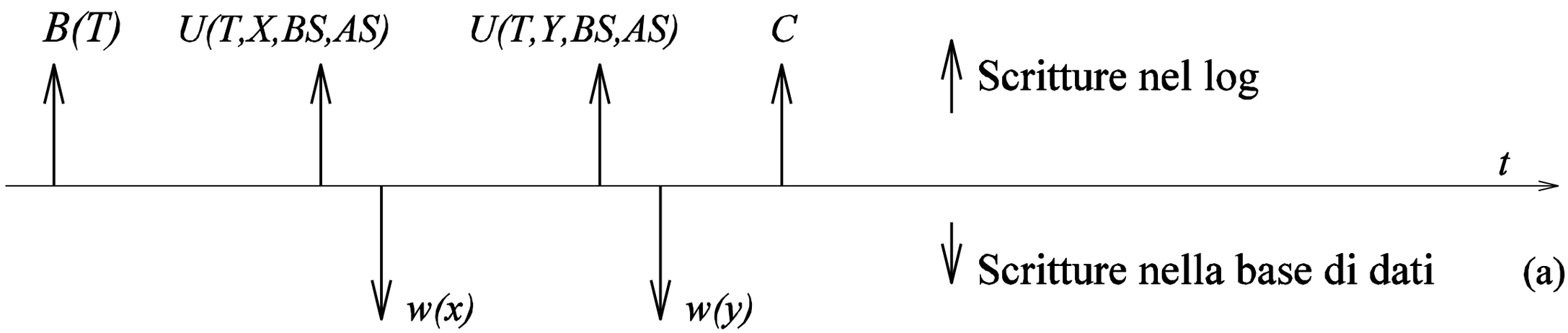
- Quando scriviamo nella base di dati?

Scrittura nel log e nella basi di dati

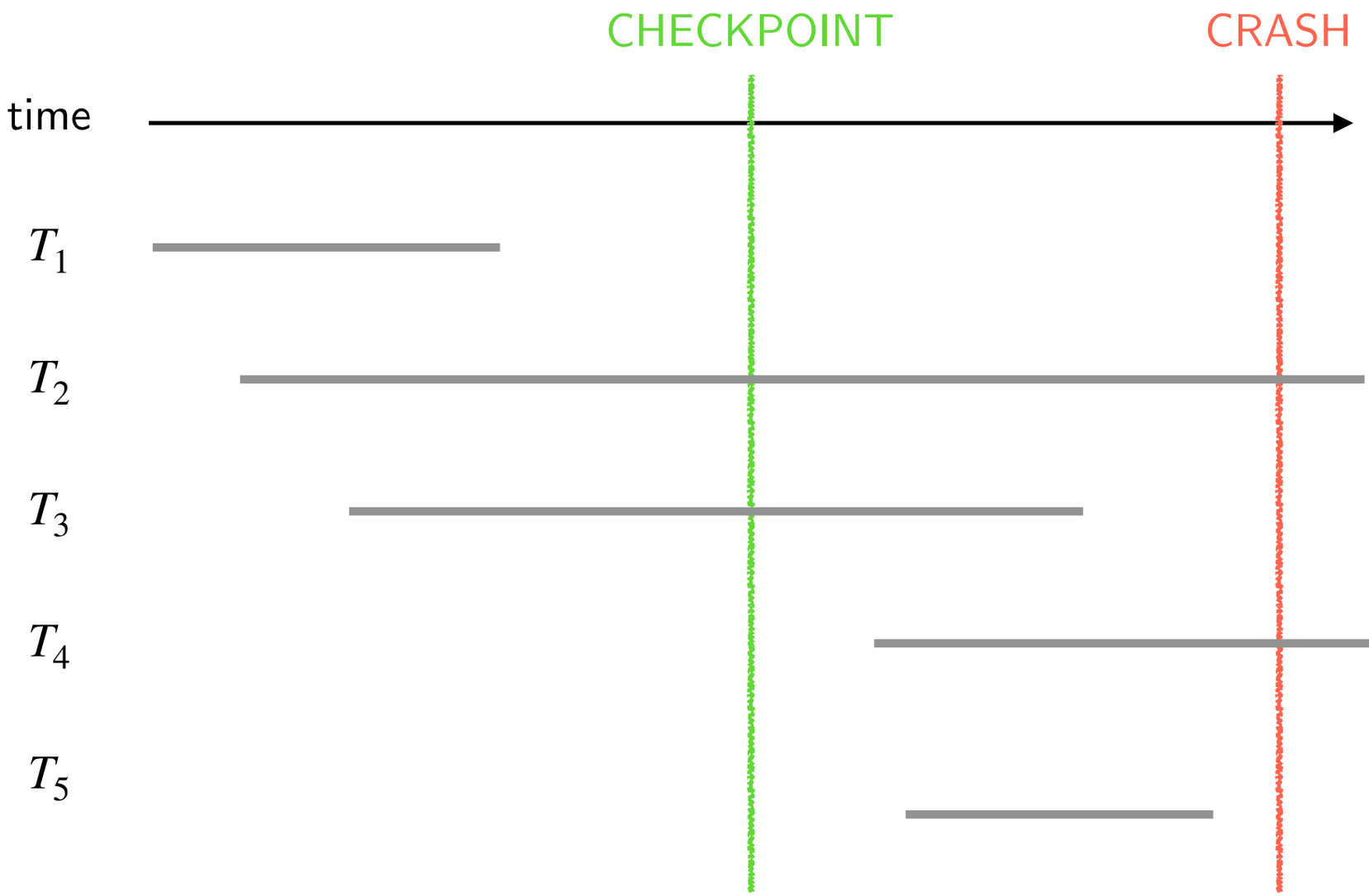


Modalità Immediata

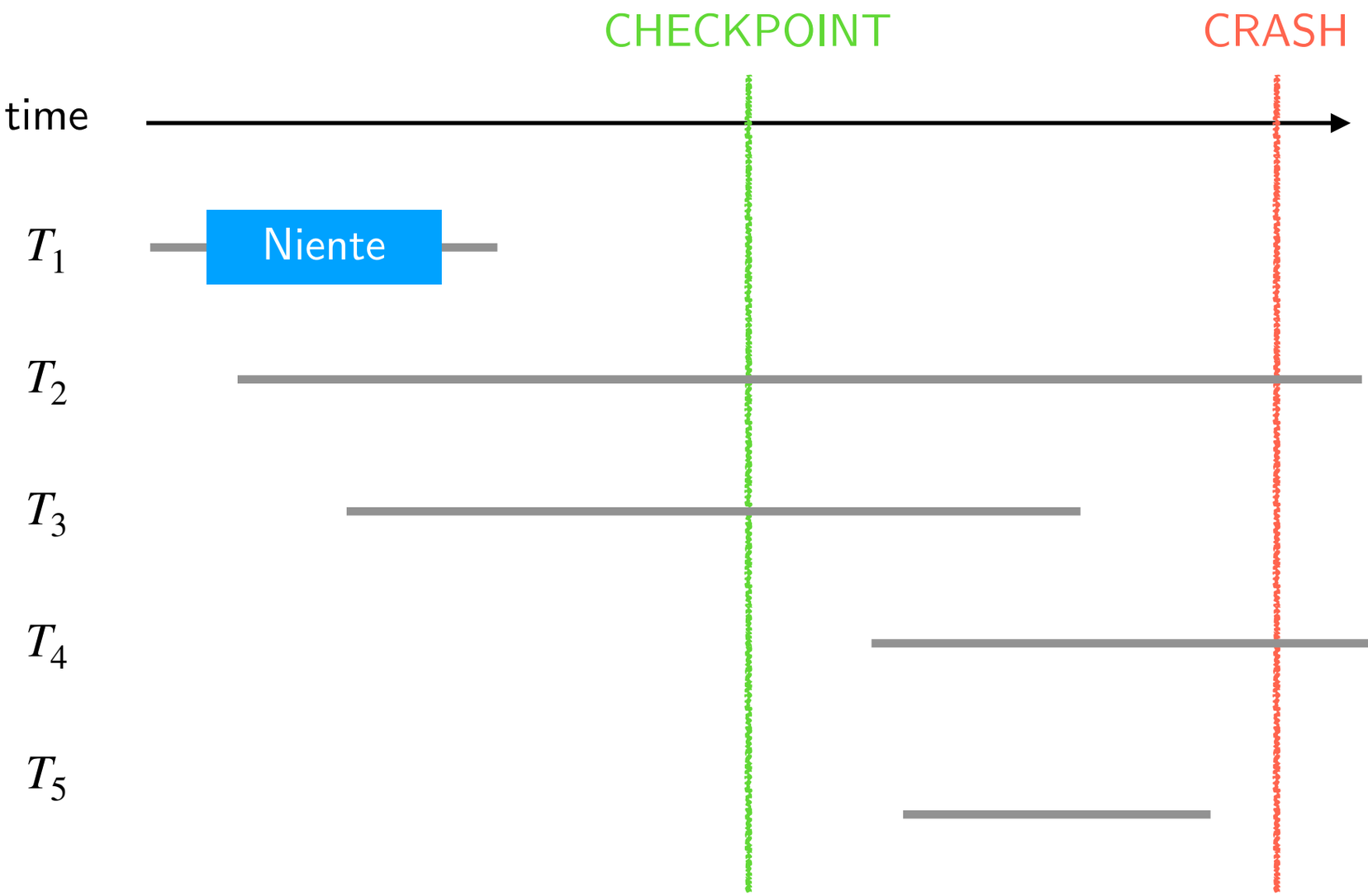
- La base di dati contiene valori AS provenienti da transazioni uncommitted
- Richiede Undo delle operazioni di transazioni uncommited al momento del guasto
- Non richiede Redo



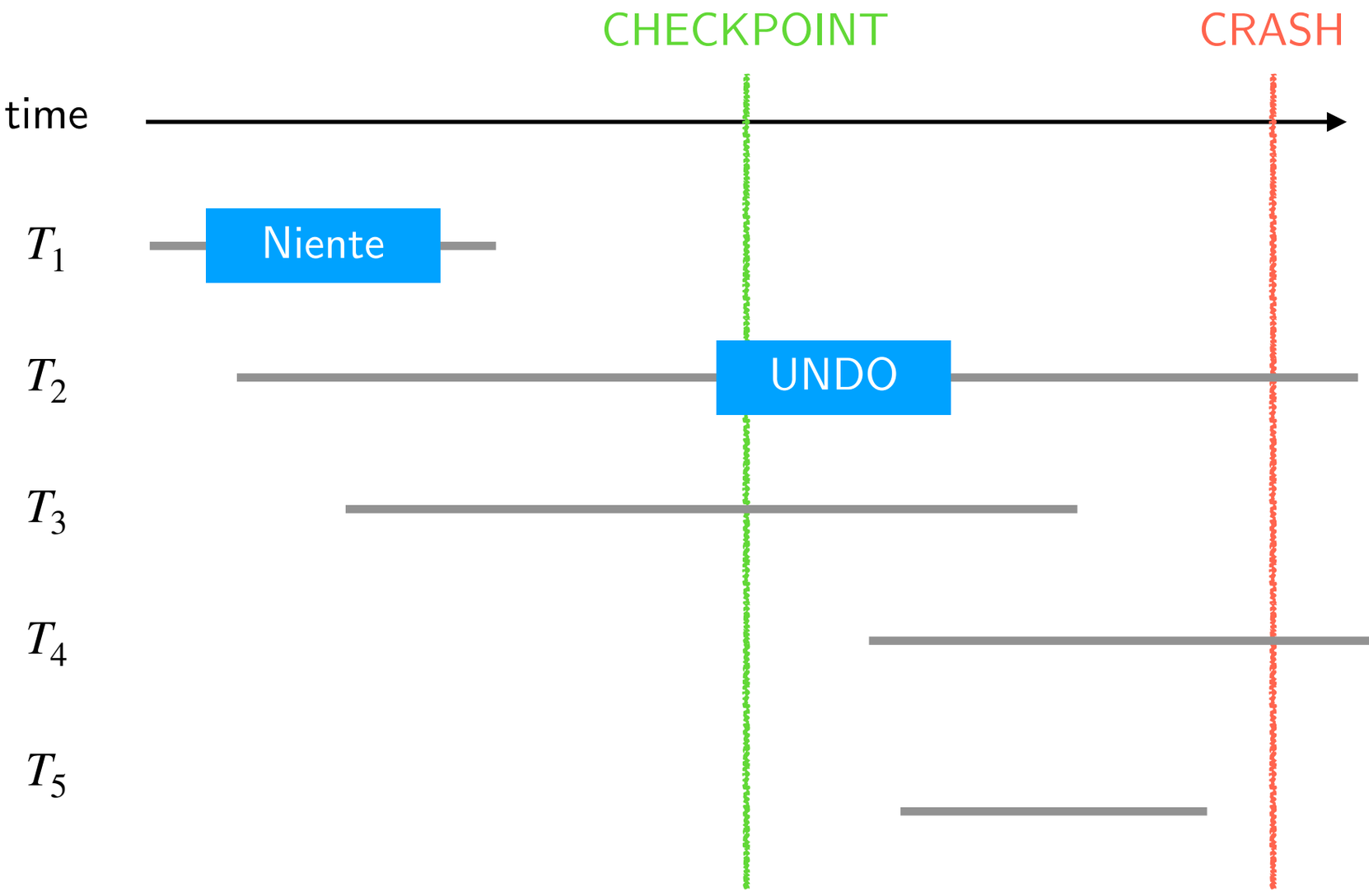
Esempio



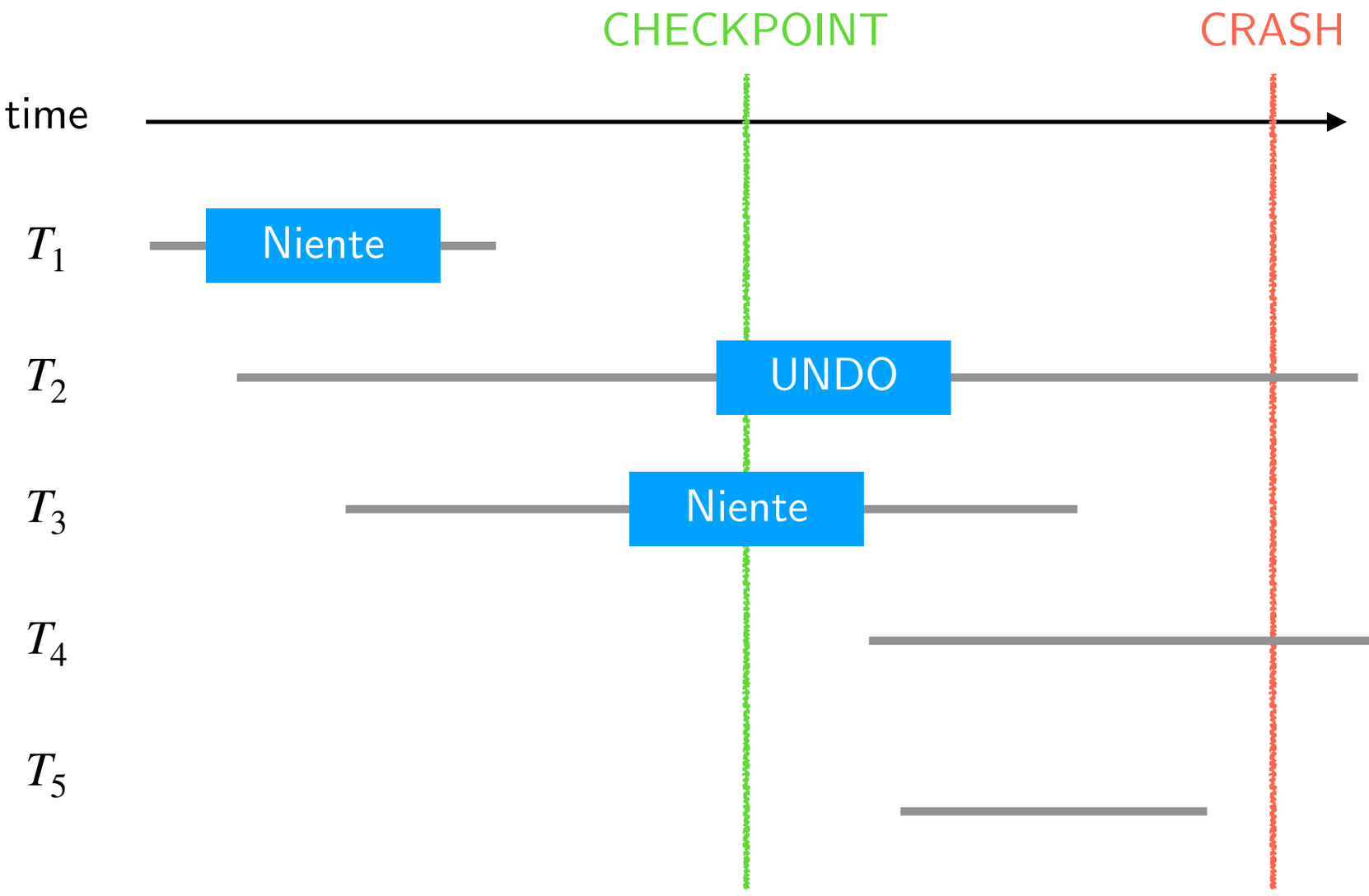
Esempio



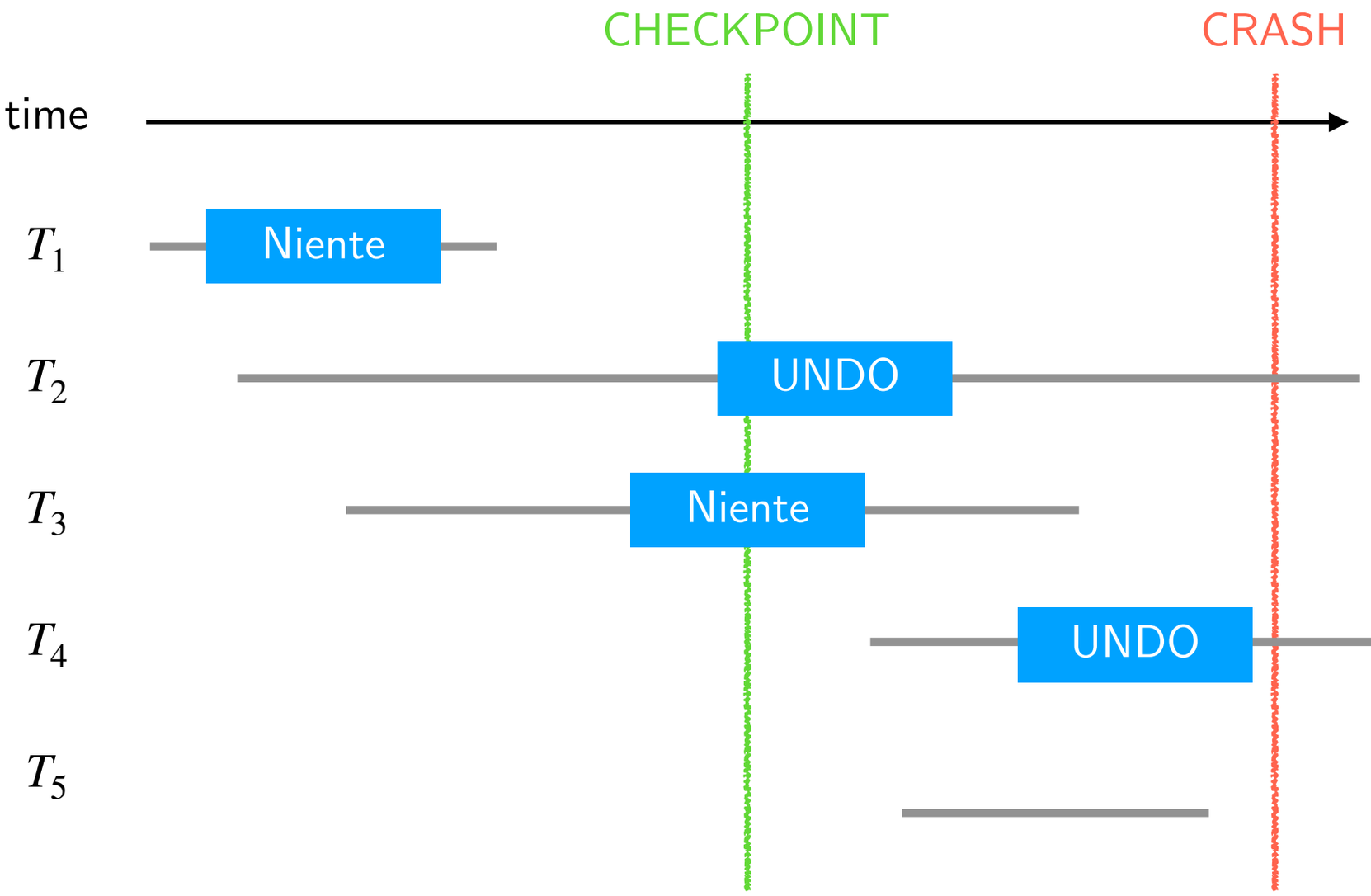
Esempio



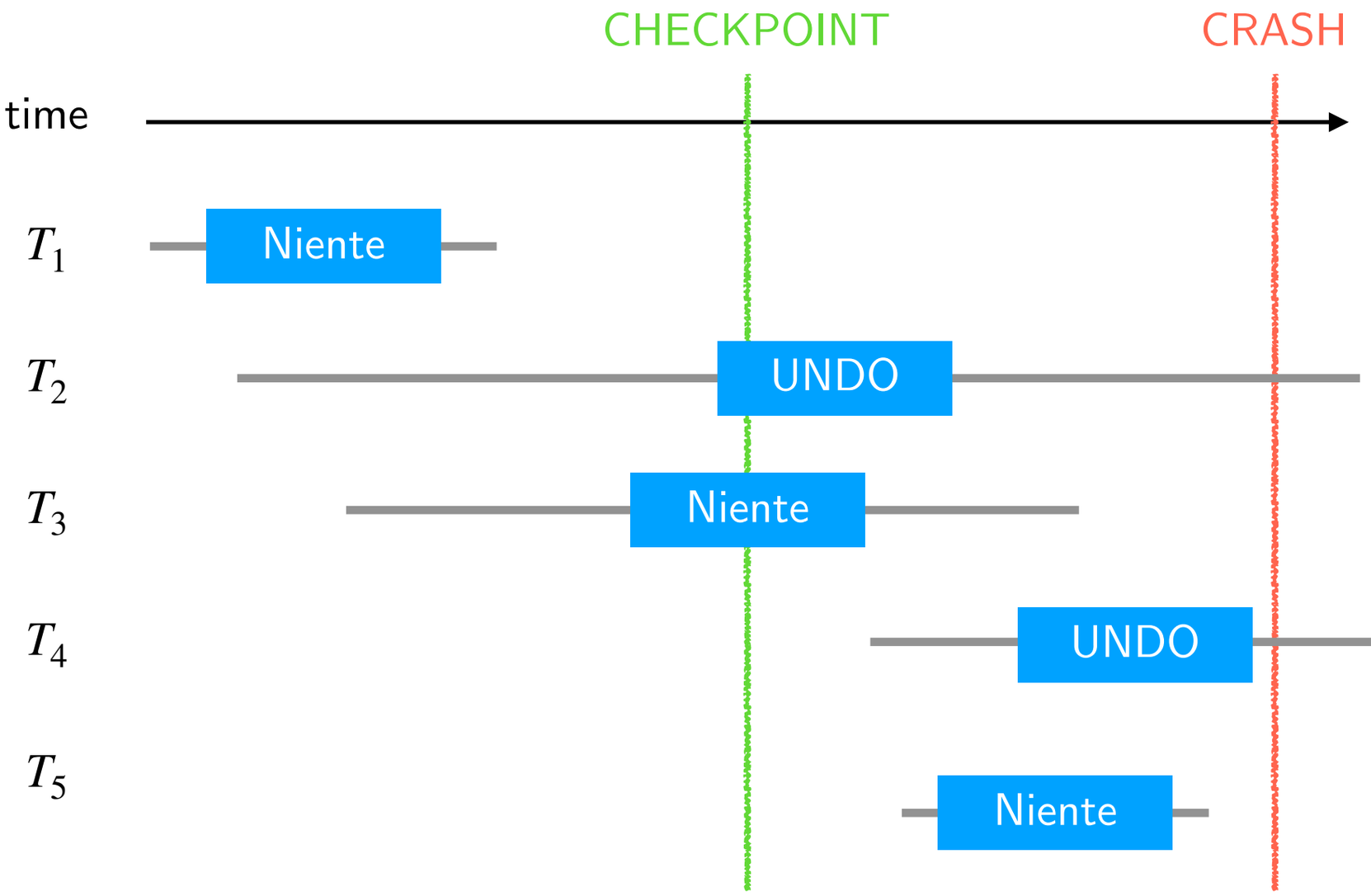
Esempio



Esempio

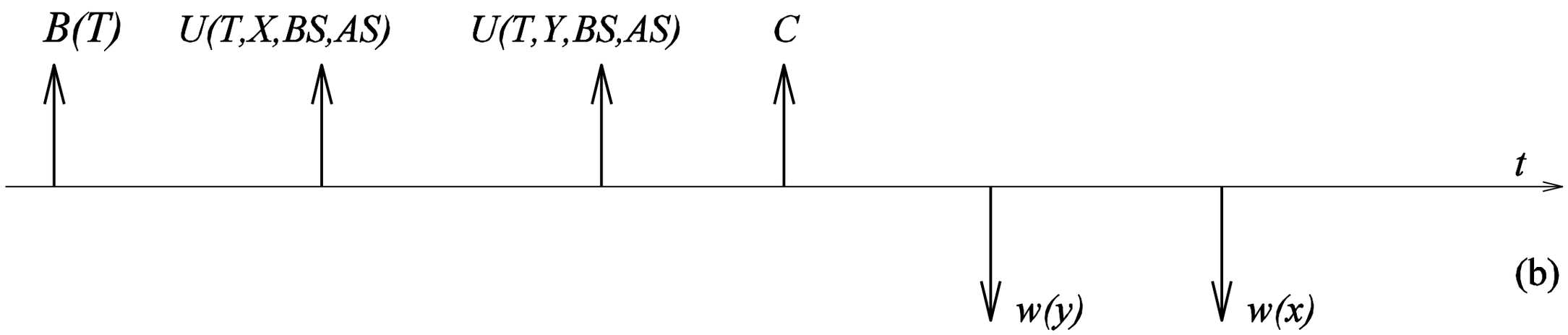


Esempio

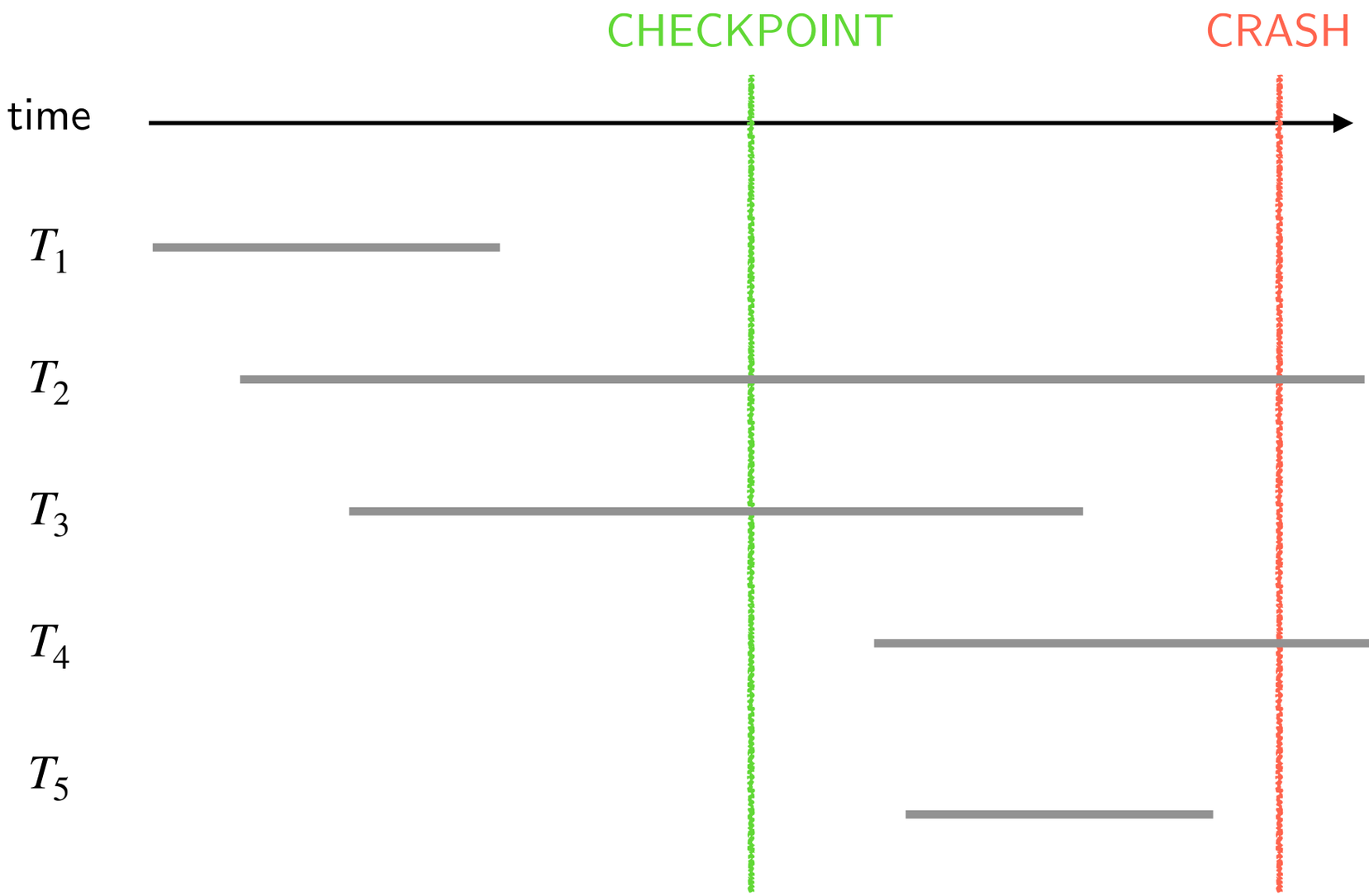

CHECKPOINT
CRASH
time
T_1
Niente
T_2
UNDO
T_3
Niente
T_4
UNDO
T_5
Niente

Modalità Differita

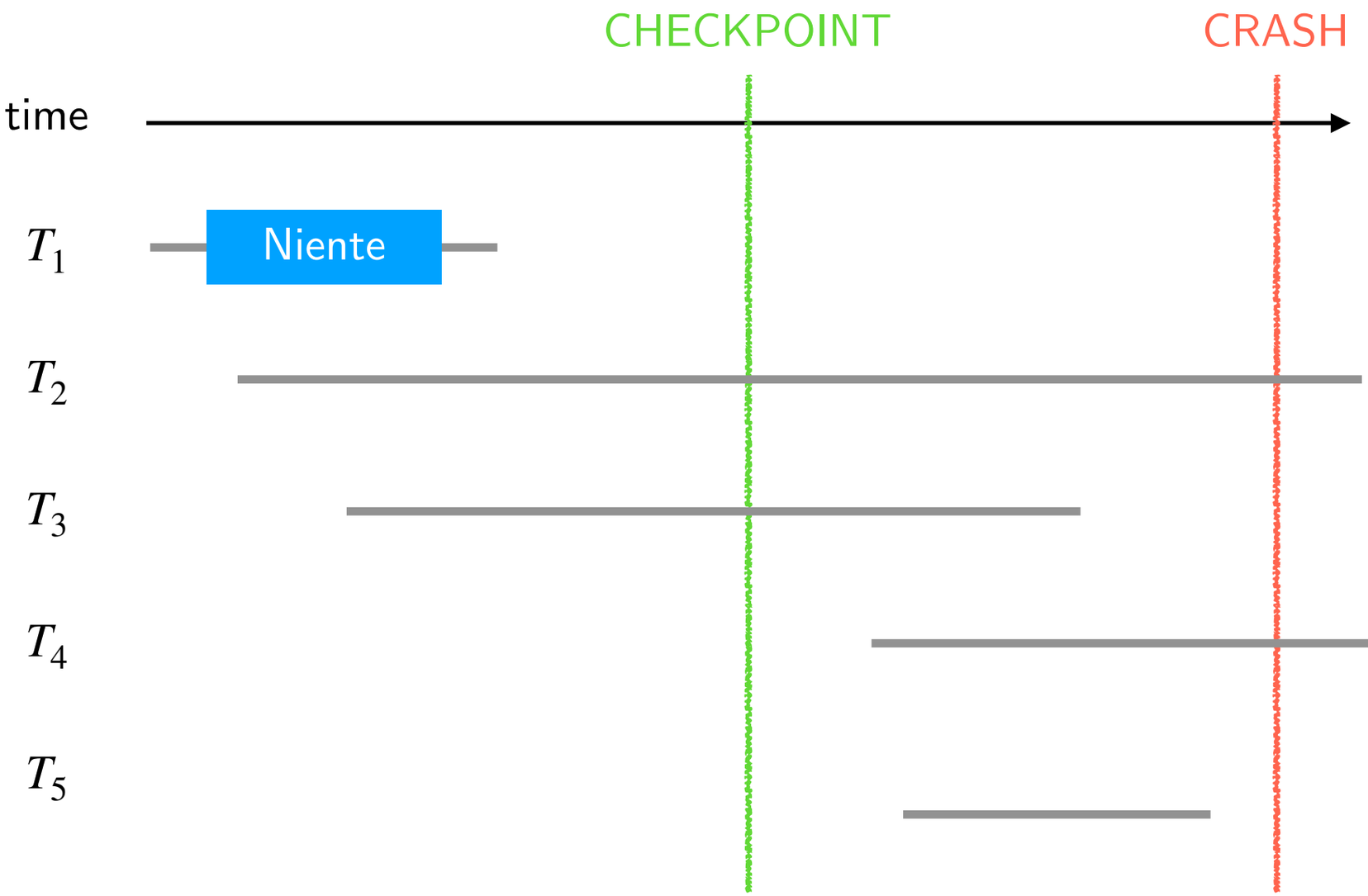
- La base di dati non contiene valori AS provenienti da transazioni uncommitted
- In caso di abort, non occorre fare niente
- Rende superflua la procedura di Undo, non ci sono scritture prima del commit
- Richiede Redo



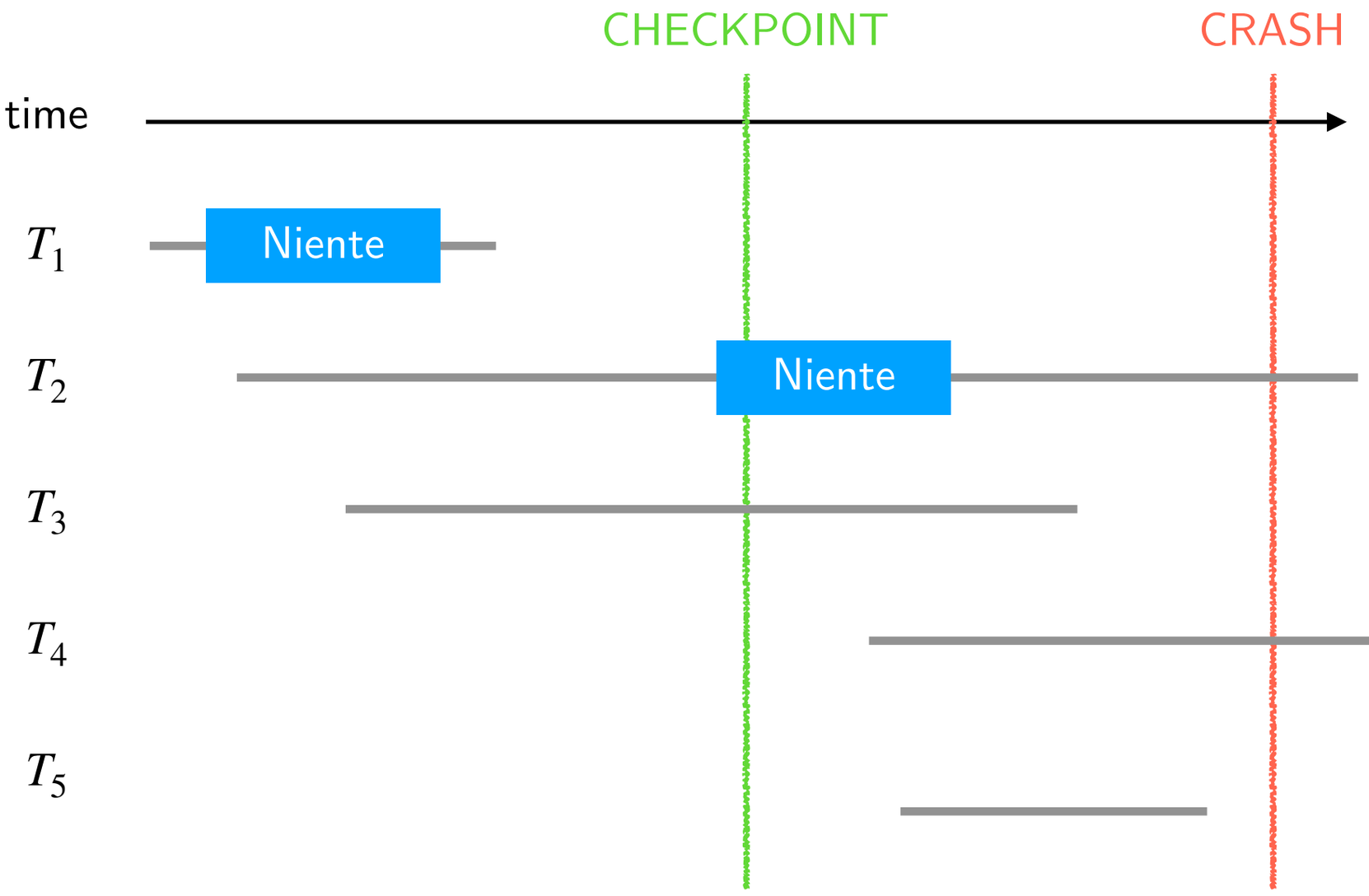
Esempio



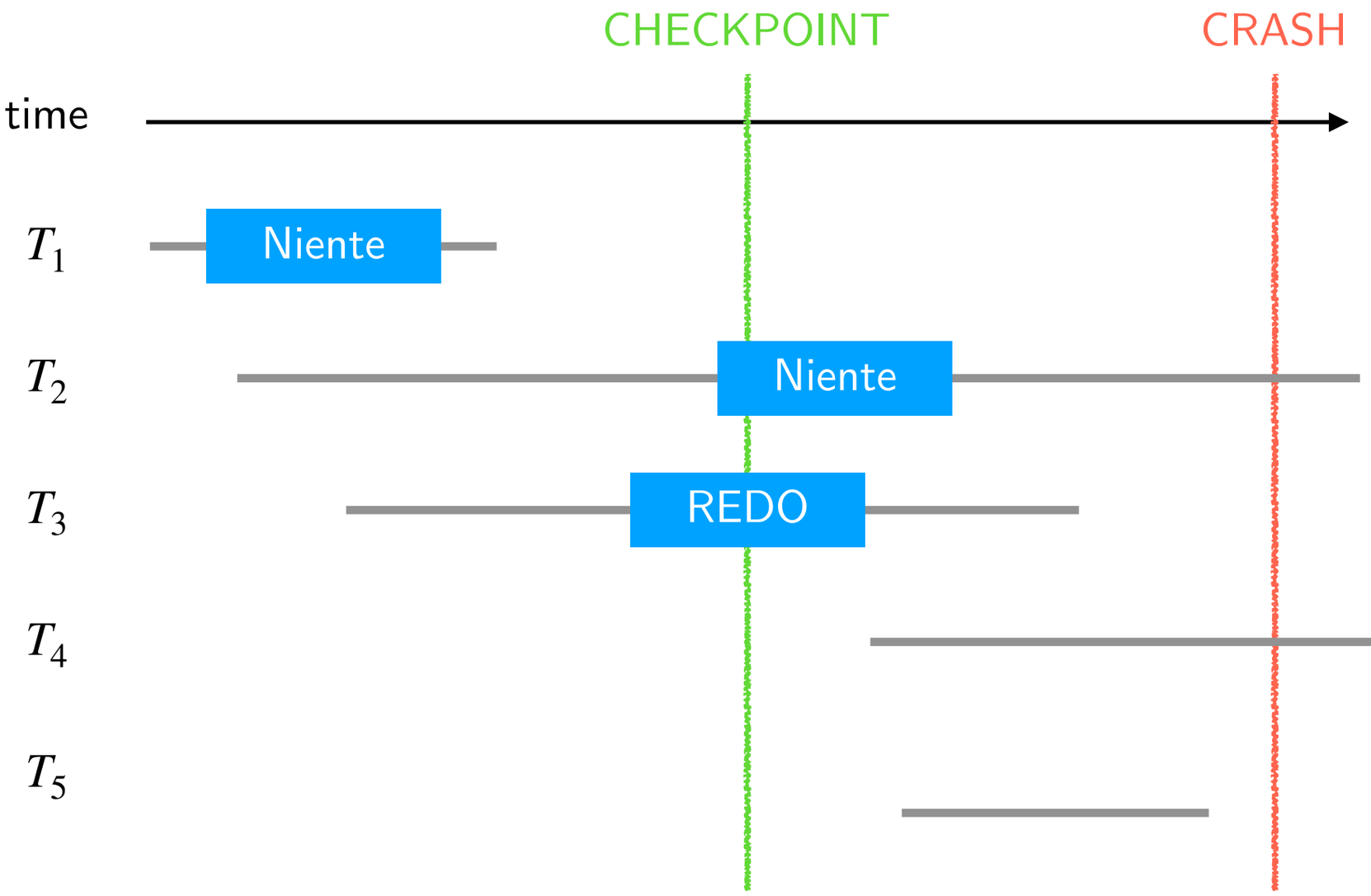
Esempio



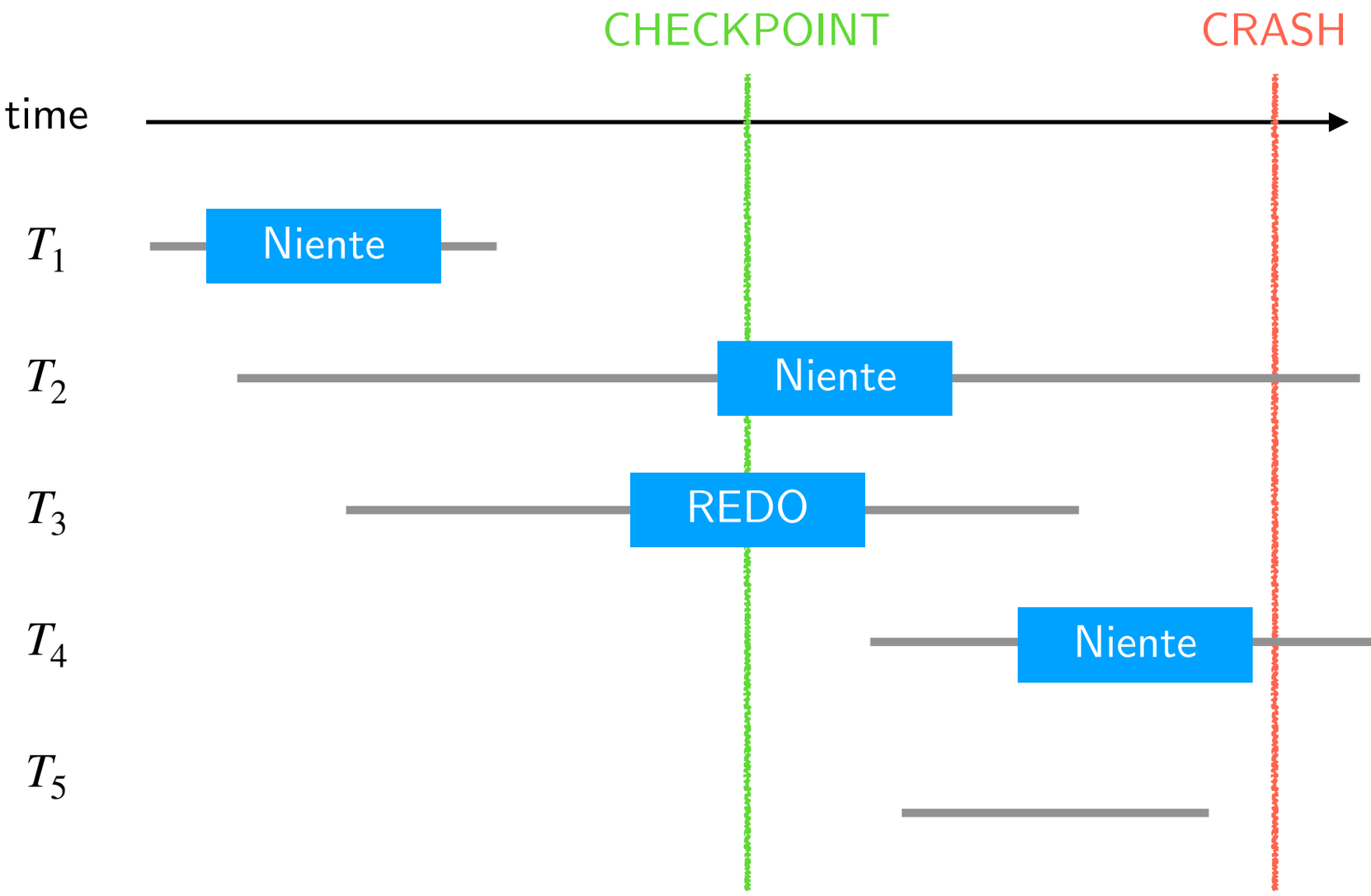
Esempio



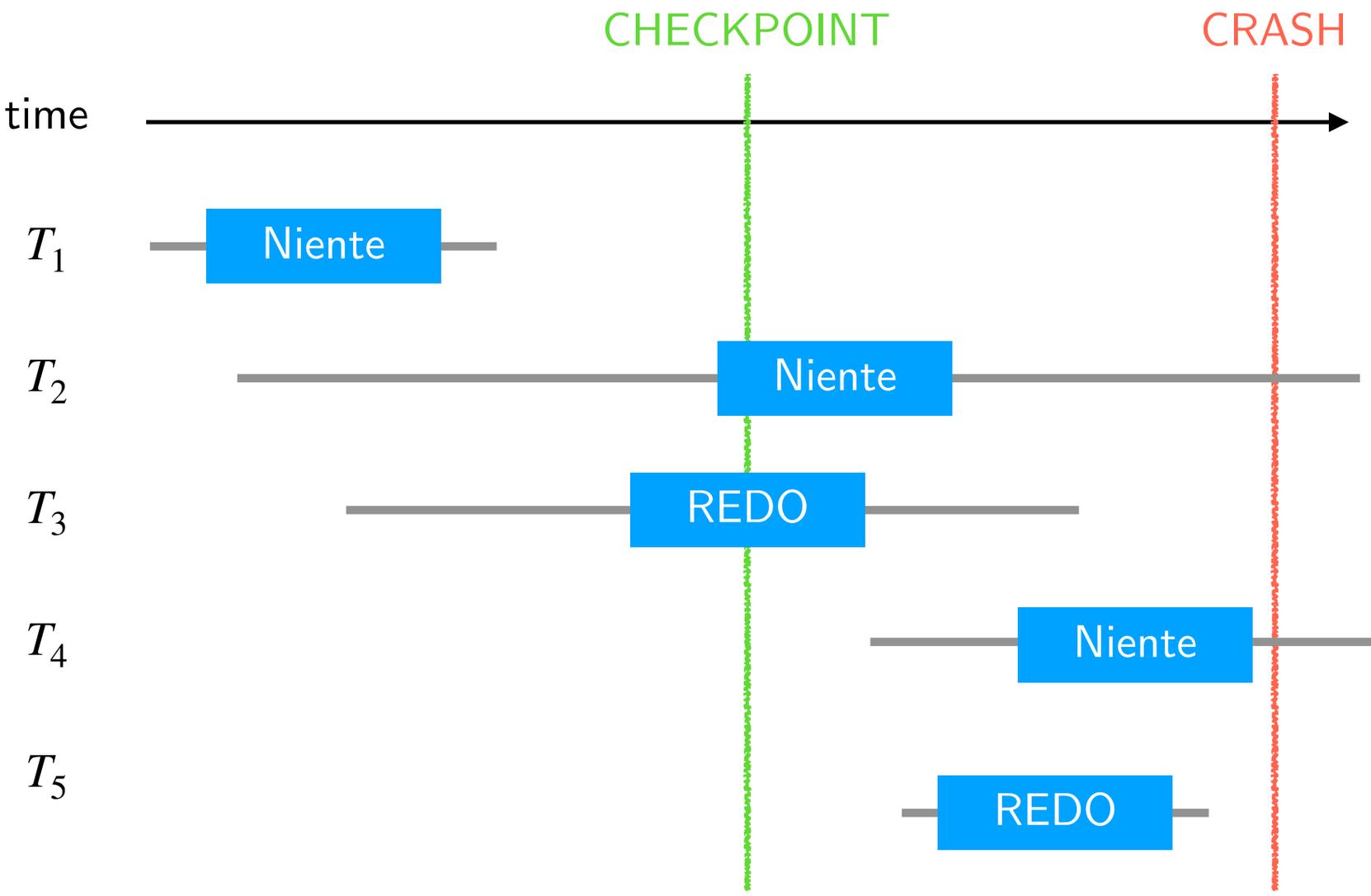
Esempio



Esempio

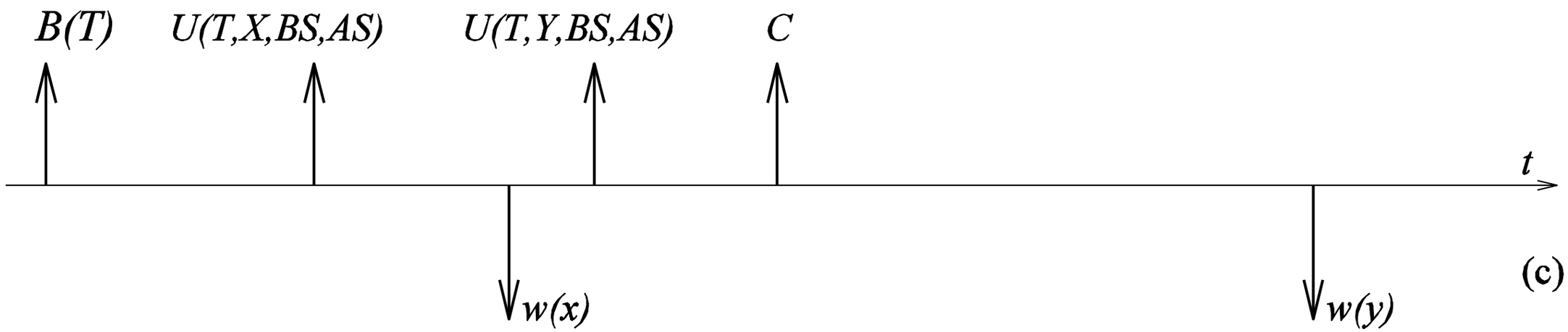

CHECKPOINT
CRASH
time
T_1
Niente
T_2
Niente
T_3
REDO
T_4
Niente
T_5

Esempio

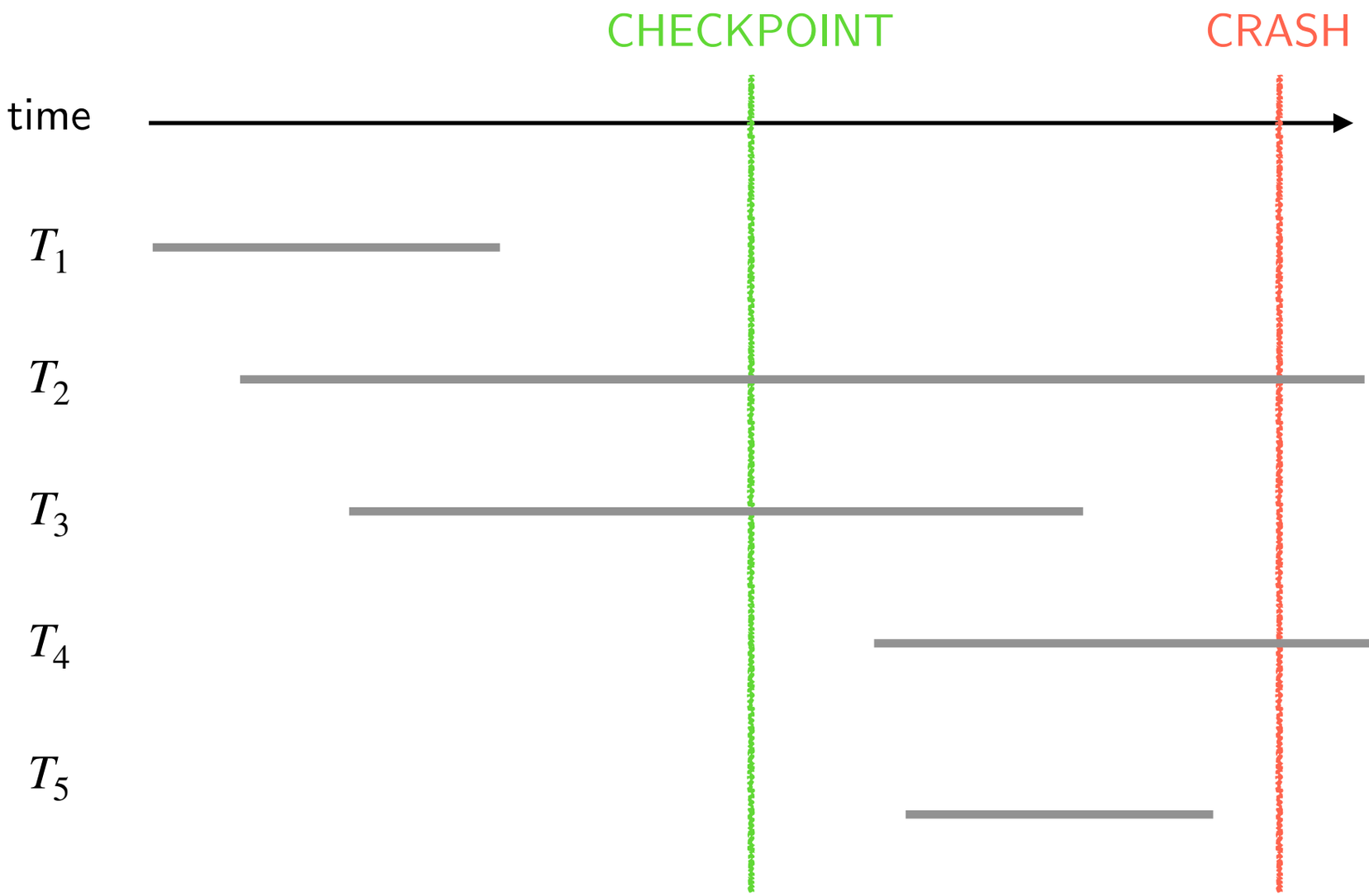

CHECKPOINT
CRASH
time
T_1
Niente
T_2
Niente
T_3
REDO
T_4
Niente
T_5
REDO

Modalità Mista

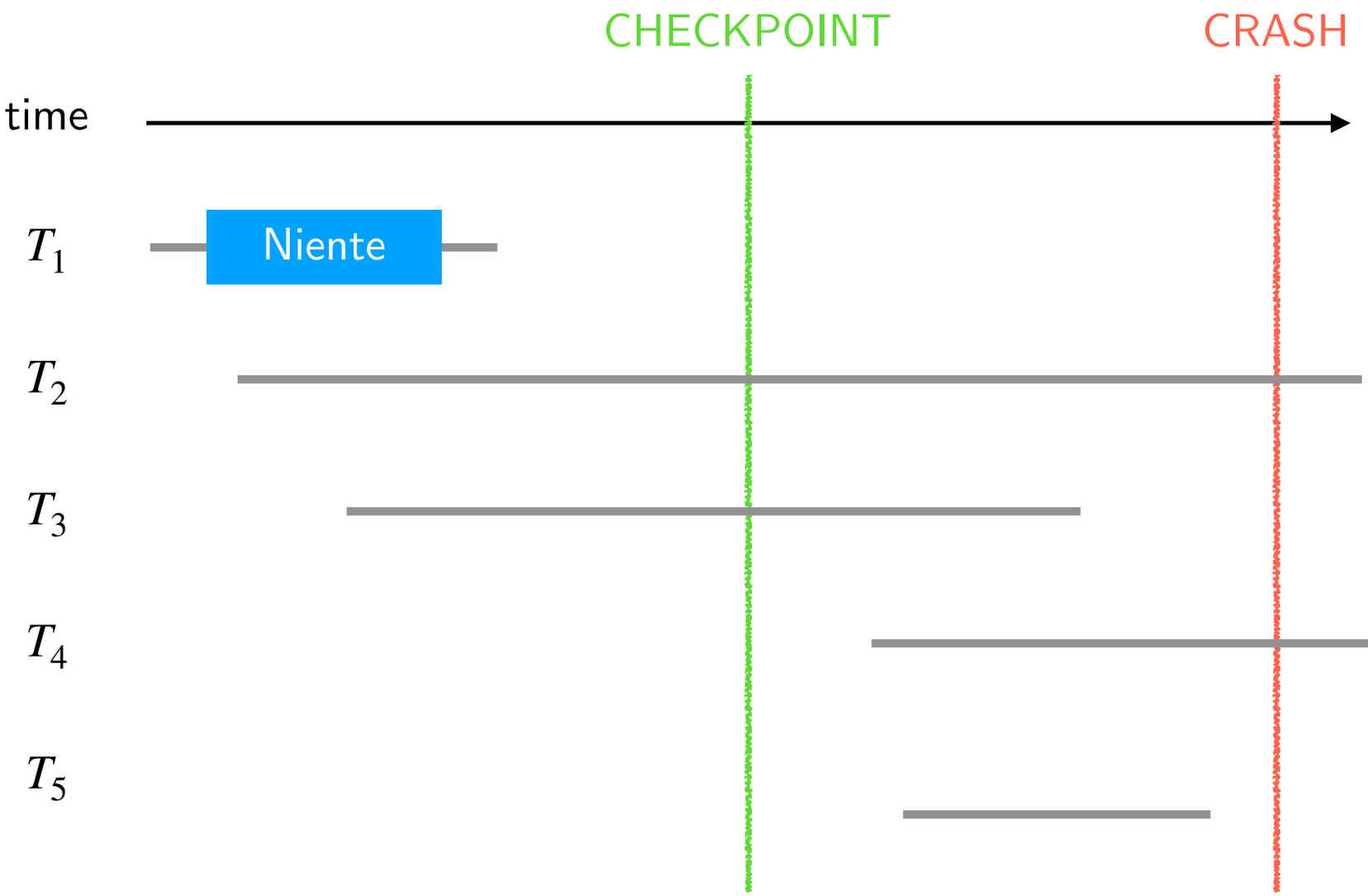
- La scrittura può avvenire in modalità sia immediata che differita
- Richiede sia Undo che Redo



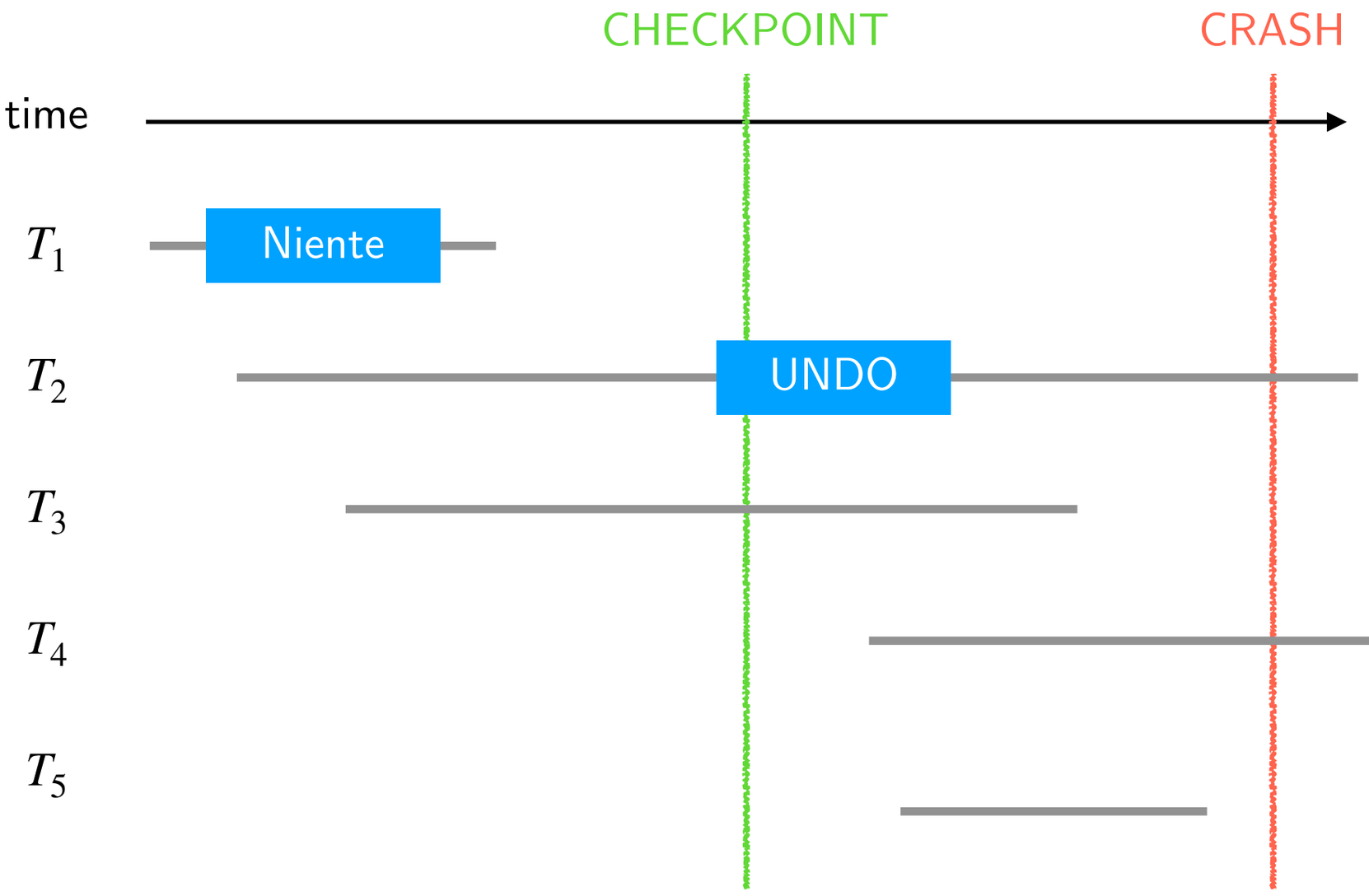
Esempio



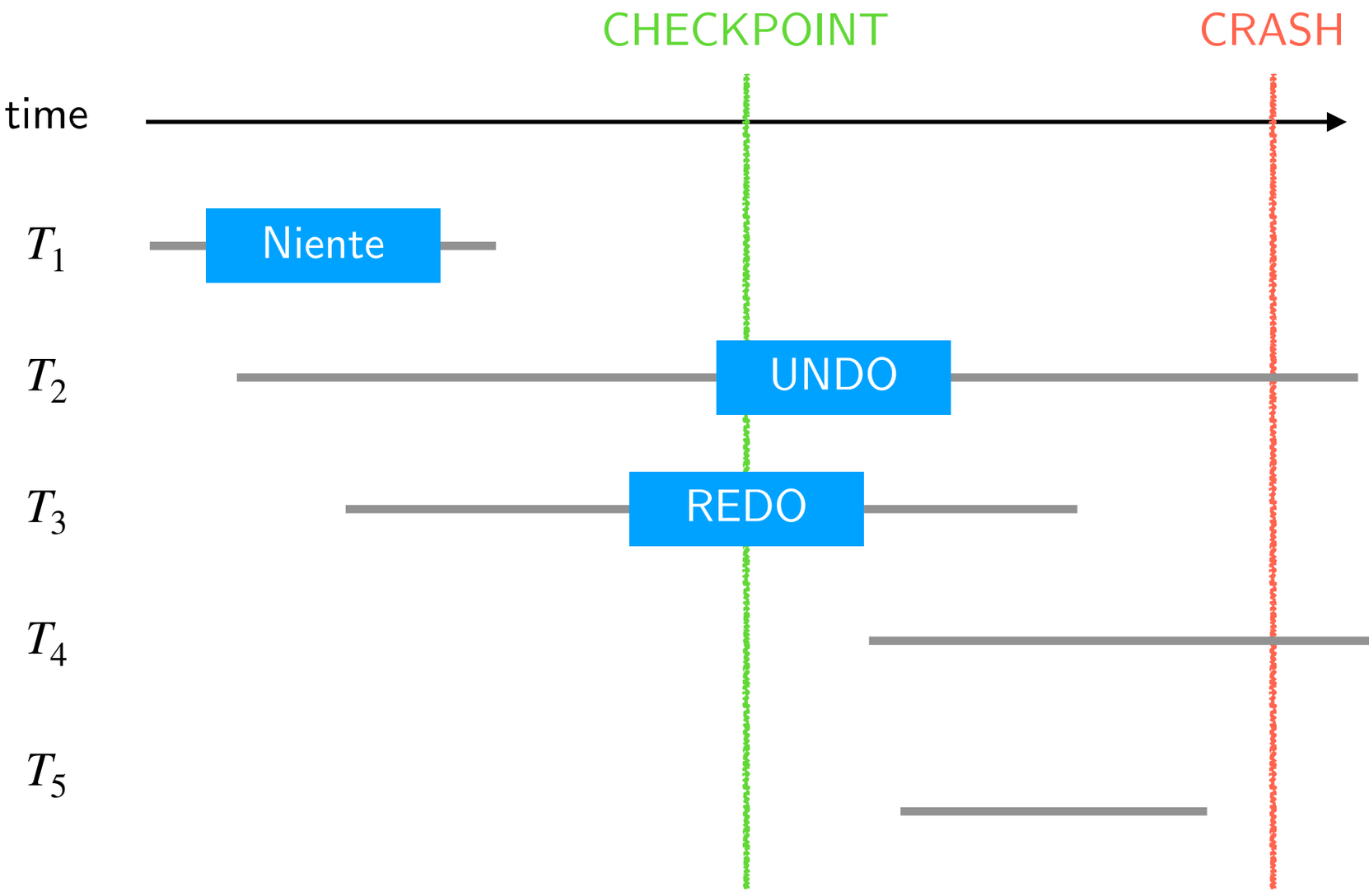
Esempio



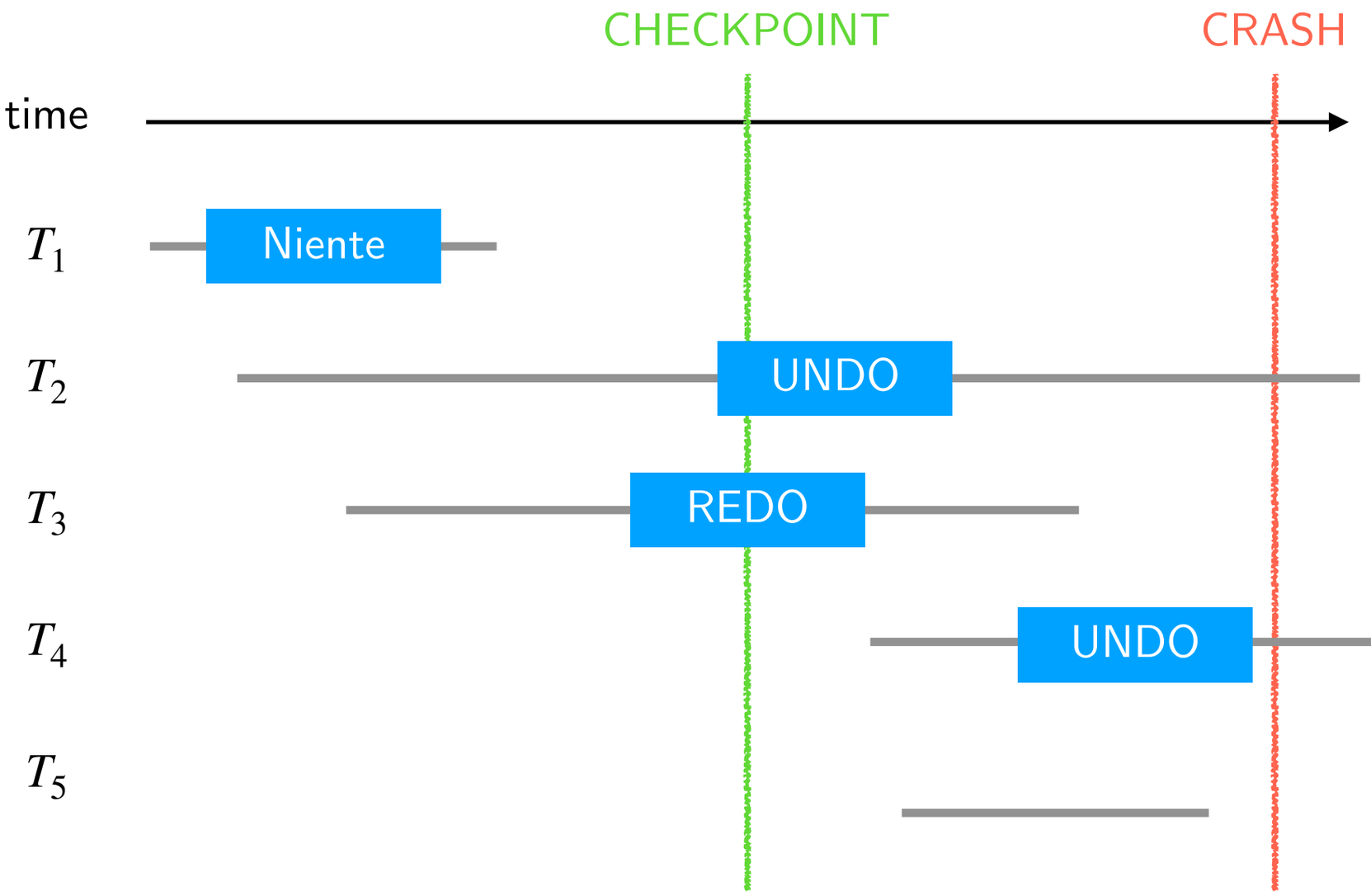
Esempio



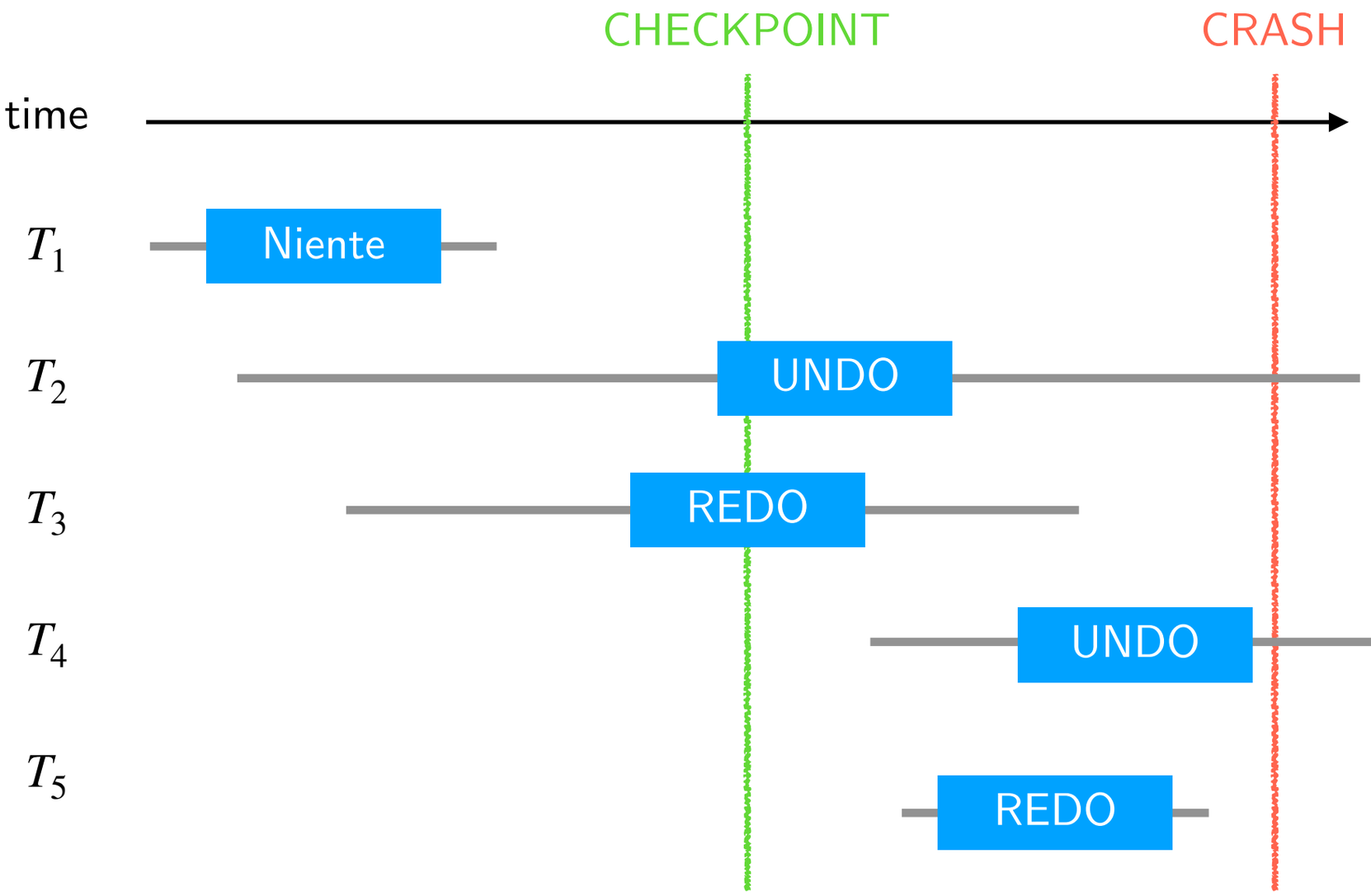
Esempio



Esempio



Esempio


CHECKPOINT
CRASH
time
T_1
Niente
T_2
UNDO
T_3
REDO
T_4
UNDO
T_5
REDO

Scrittura nel log e nella basi di dati

Scrittura nel log e nella basi di dati

- La **modalità differita** di scrittura, pur permettendo una procedura di recupero più semplice ed efficiente, **non** viene **molto utilizzata in pratica**. Perché?

Scrittura nel log e nella basi di dati

- La **modalità differita** di scrittura, pur permettendo una procedura di recupero più semplice ed efficiente, **non** viene **molto utilizzata in pratica**. Perché?
- Questa modalità è **più efficiente** nel **recovery**, ma è complessivamente **meno efficiente** di una in cui il gestore può decidere liberamente quando **scrivere** in **memoria** secondaria. Quindi?

Scrittura nel log e nella basi di dati

- La **modalità differita** di scrittura, pur permettendo una procedura di recupero più semplice ed efficiente, **non** viene **molto utilizzata in pratica**. Perché?
- Questa modalità è **più efficiente** nel **recovery**, ma è complessivamente **meno efficiente** di una in cui il gestore può decidere liberamente quando **scrivere** in **memoria** secondaria. Quindi?
- È preferibile una gestione ordinaria più efficiente rispetto ad una gestione più semplice dei guasti, poiché **si assume che i guasti siano abbastanza rari**.

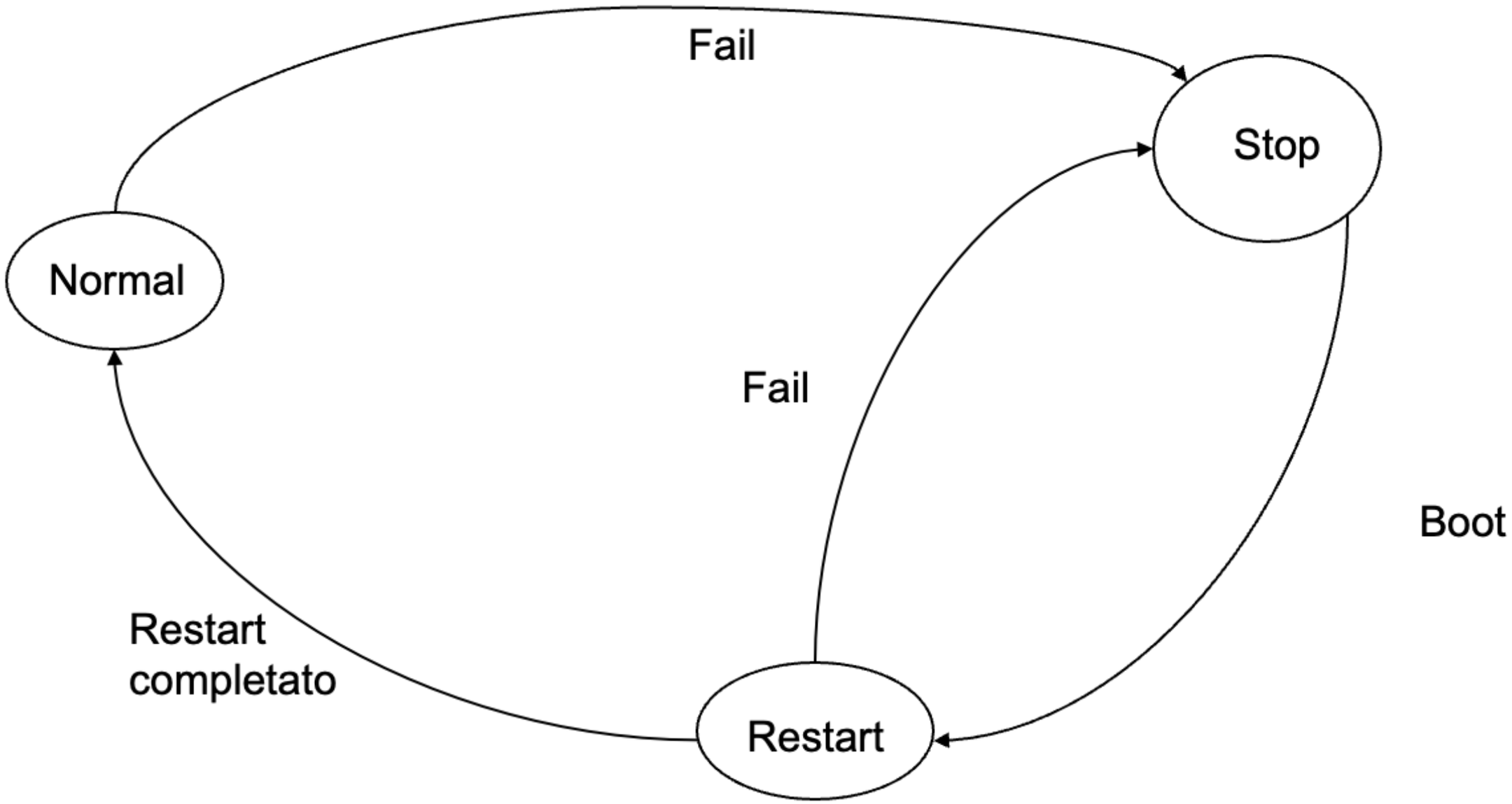
Guasti

- **Guasti "soft"**: errori di programma, crash di sistema, caduta di tensione
  - si **perde** la **memoria centrale**
  - **non** si **perde** la **memoria secondaria**, cioè la base di dati
  - **non** si **perde** la **memoria stabile**, cioè il log
  - ***warm restart***, **ripresa a caldo**
- **Guasti "hard"**: dei dispositivi di memoria secondaria
  - si **perde** anche la **memoria secondaria**, cioè parte della base di dati
  - **non** si **perde** la **memoria stabile**, cioè il log
  - ***cold restart***, **ripresa a freddo**
- La **perdita del log** è considerato un **evento catastrofico** e quindi non è definita alcuna strategia di recupero

Modello di funzionamento Fail-Stop

- L'individuazione di un **guasto** forza **l'arresto completo** delle transazioni
- Il sistema operativo viene **riavviato**
- Viene avviata una procedura di ***restart***
- Al termine del *restart* il **buffer** è vuoto, ma le transazioni possono ripartire

Modello di funzionamento Fail-Stop



Processo di restart

- Obiettivo: **classificare le transazioni** in
  - **completate** (tutti i dati in memoria stabile)
  - in **commit** ma **non necessariamente completate** (può servire REDO)
  - **senza commit** (vanno annullate, UNDO)

Protocollo del gestore dell'affidabilità

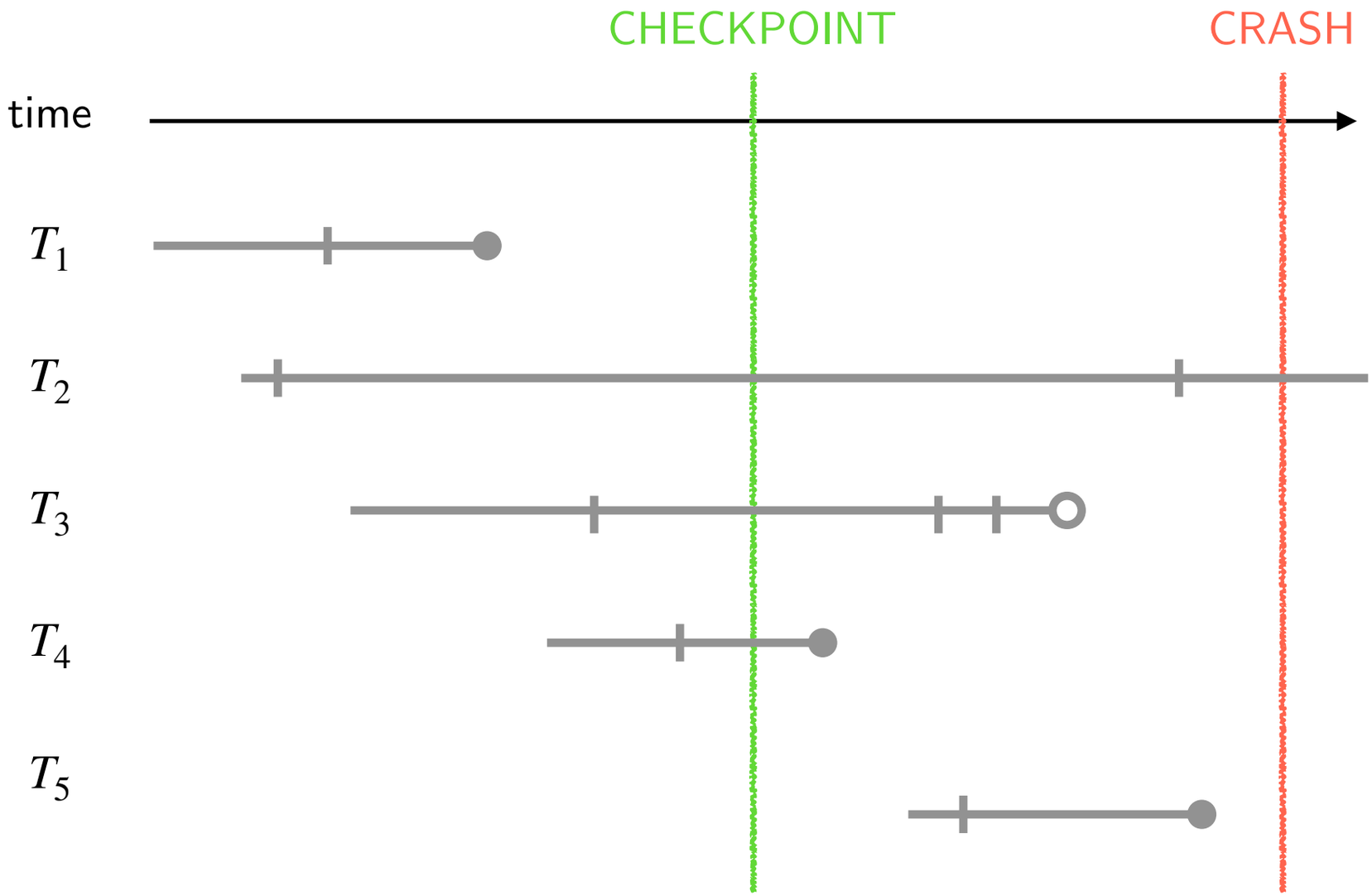
- Il gestore dell'affidabilità, al restart del sistema,
  1. **Legge su un file** di RESTART (sempre contenuto nel log) l'indirizzo dell'ultimo checkpoint.
  2. **Prepara due file**: **UNDO list** con gli identificatori delle transazioni attive, **REDO list** vuoto
  3. **Nessun utente è attivo** durante il RESTART

Ripresa a caldo

- **Quattro** fasi:
  - **trovare l'ultimo checkpoint** (ripercorrendo il log a ritroso)
  - **costruire** gli insiemi **UNDO** (transazioni attive ma non committed prima del guasto, da disfare) e **REDO** (transazioni committed tra il CK e il guasto, da rifare)
- **ripercorrere** il **log all'indietro** (***rollback***), fino alla più vecchia azione delle transazioni in UNDO e REDO, **disfacendo** tutte le azioni delle **transazioni** in **UNDO**
- **ripercorrere** il **log in avanti** (***rollforward***), **rifacendo** tutte le azioni delle **transazioni** in **REDO**

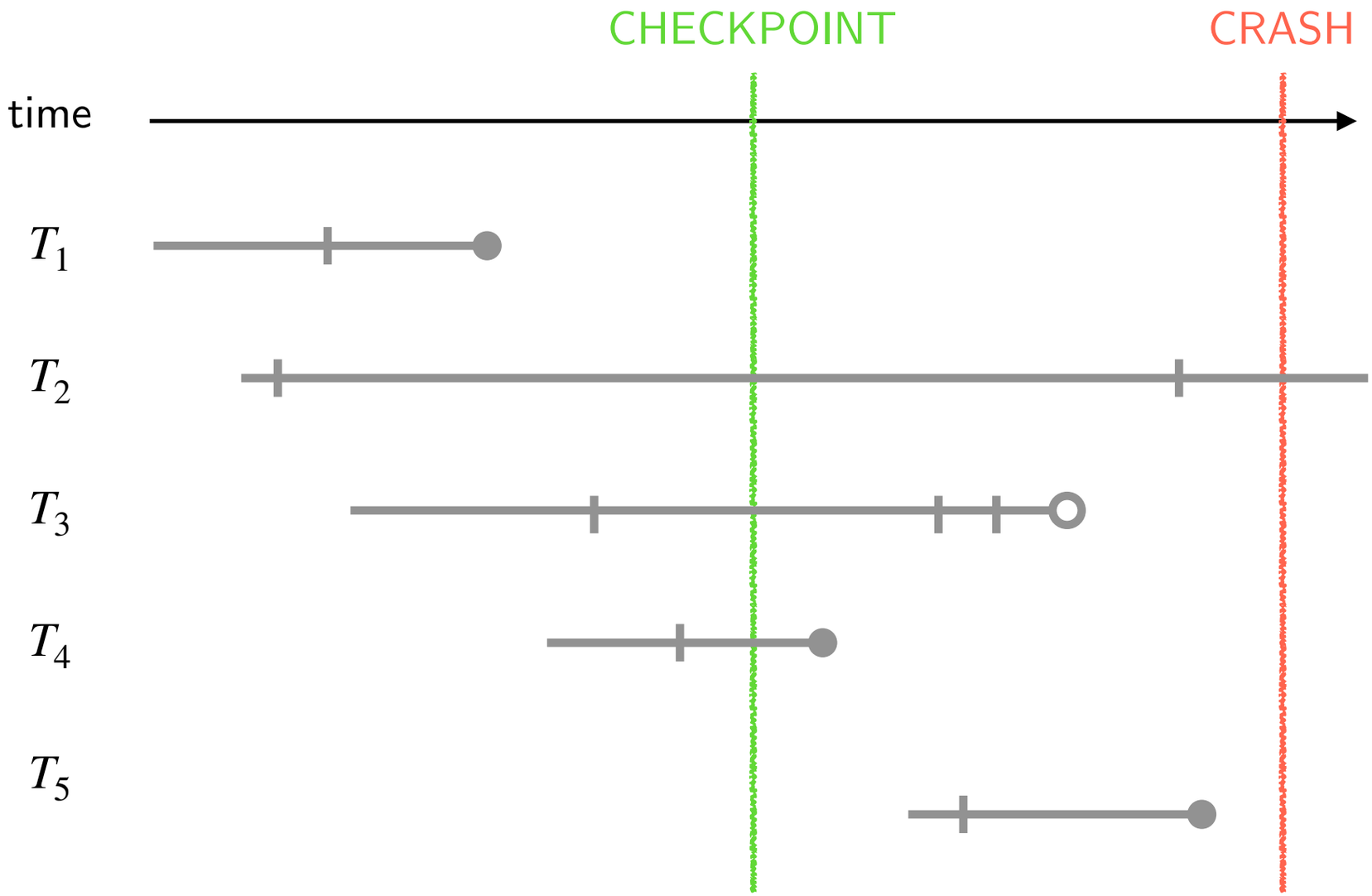
Esempio

$B(T_1)$
$B(T_2)$
$U(T_2, O_1, B_1, A_1)$
$I(T_1, O_2, A_2)$
$B(T_3)$
$C(T_1)$
$B(T_4)$
$U(T_3, O_2, B_3, A_3)$
$U(T_4, O_3, B_4, A_4)$
$CK(T_2, T_3, T_4)$
$C(T_4)$
$B(T_5)$
$U(T_3, O_3, B_5, A_5)$
$U(T_5, O_4, B_6, A_6)$
$D(T_3, O_5, B_7)$
$A(T_3)$
$C(T_5)$
$I(T_2, O_6, A_8)$



Trovare l'ultimo checkpoint

$B(T_1)$
$B(T_2)$
$U(T_2, O_1, B_1, A_1)$
$I(T_1, O_2, A_2)$
$B(T_3)$
$C(T_1)$
$B(T_4)$
$U(T_3, O_2, B_3, A_3)$
$U(T_4, O_3, B_4, A_4)$
$CK(T_2, T_3, T_4)$
$C(T_4)$
$B(T_5)$
$U(T_3, O_3, B_5, A_5)$
$U(T_5, O_4, B_6, A_6)$
$D(T_3, O_5, B_7)$
$A(T_3)$
$C(T_5)$
$I(T_2, O_6, A_8)$

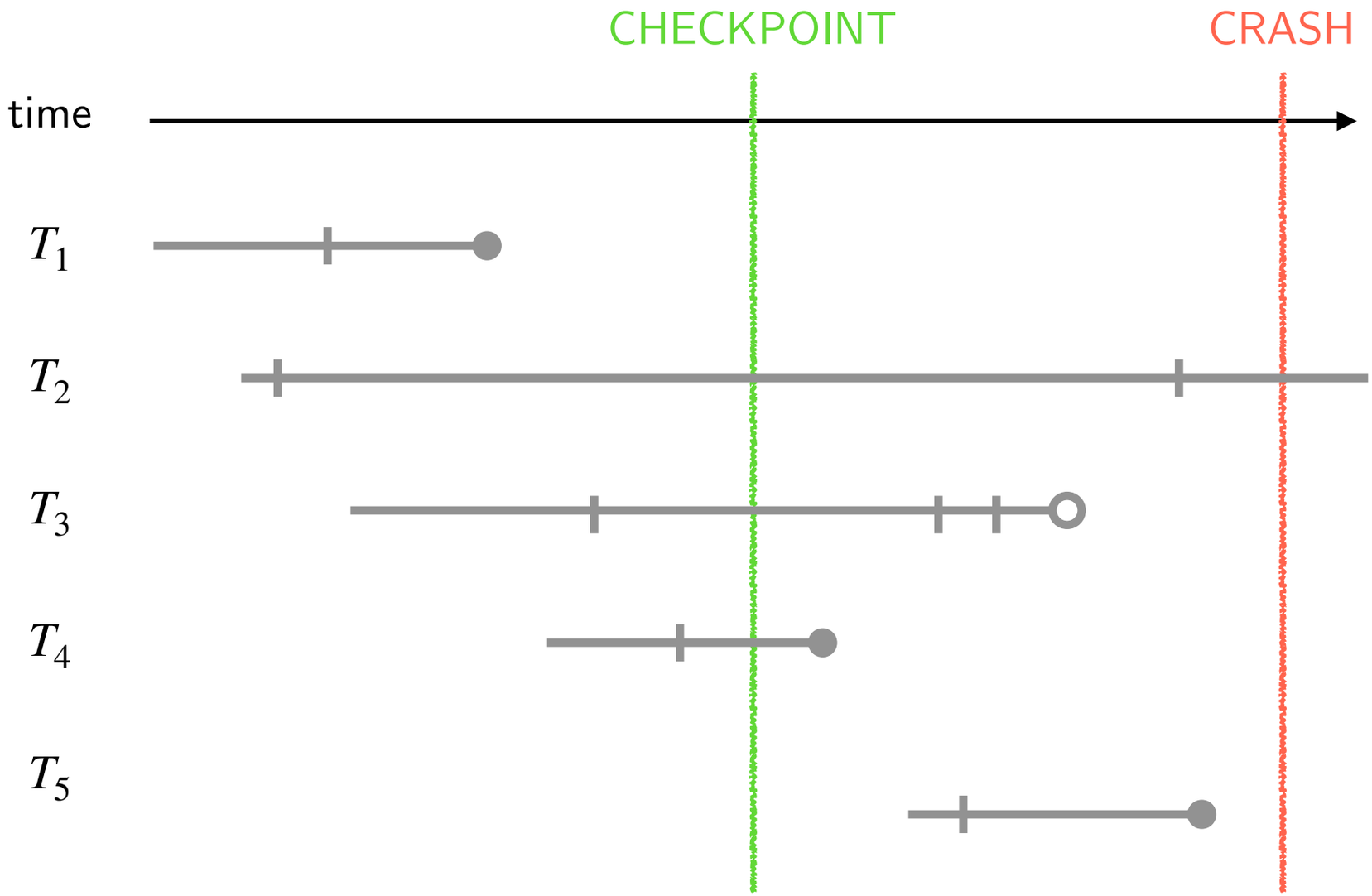


UNDO $= \{T_2, T_3, T_4\}$

REDO $= \{\}$

Costruire UNDO e REDO

$B(T_1)$
$B(T_2)$
$U(T_2, O_1, B_1, A_1)$
$I(T_1, O_2, A_2)$
$B(T_3)$
$C(T_1)$
$B(T_4)$
$U(T_3, O_2, B_3, A_3)$
$U(T_4, O_3, B_4, A_4)$
$CK(T_2, T_3, T_4)$
$C(T_4)$
$B(T_5)$
$U(T_3, O_3, B_5, A_5)$
$U(T_5, O_4, B_6, A_6)$
$D(T_3, O_5, B_7)$
$A(T_3)$
$C(T_5)$
$I(T_2, O_6, A_8)$

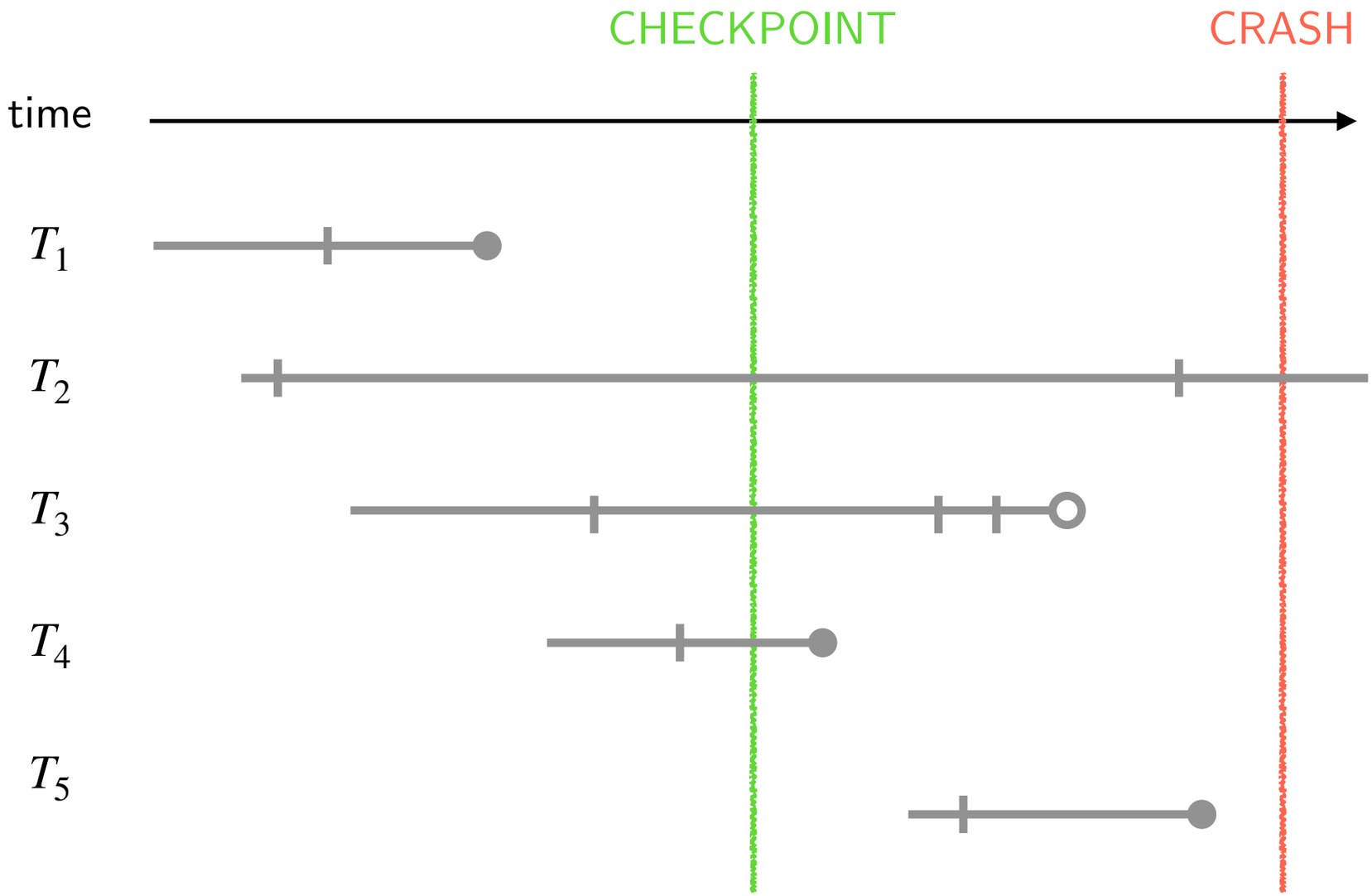


UNDO $= \{T_2, T_3\}$

REDO $= \{T_4\}$

Costruire UNDO e REDO

$B(T_1)$
$B(T_2)$
$U(T_2, O_1, B_1, A_1)$
$I(T_1, O_2, A_2)$
$B(T_3)$
$C(T_1)$
$B(T_4)$
$U(T_3, O_2, B_3, A_3)$
$U(T_4, O_3, B_4, A_4)$
$CK(T_2, T_3, T_4)$
$C(T_4)$
$B(T_5)$
$U(T_3, O_3, B_5, A_5)$
$U(T_5, O_4, B_6, A_6)$
$D(T_3, O_5, B_7)$
$A(T_3)$
$C(T_5)$
$I(T_2, O_6, A_8)$

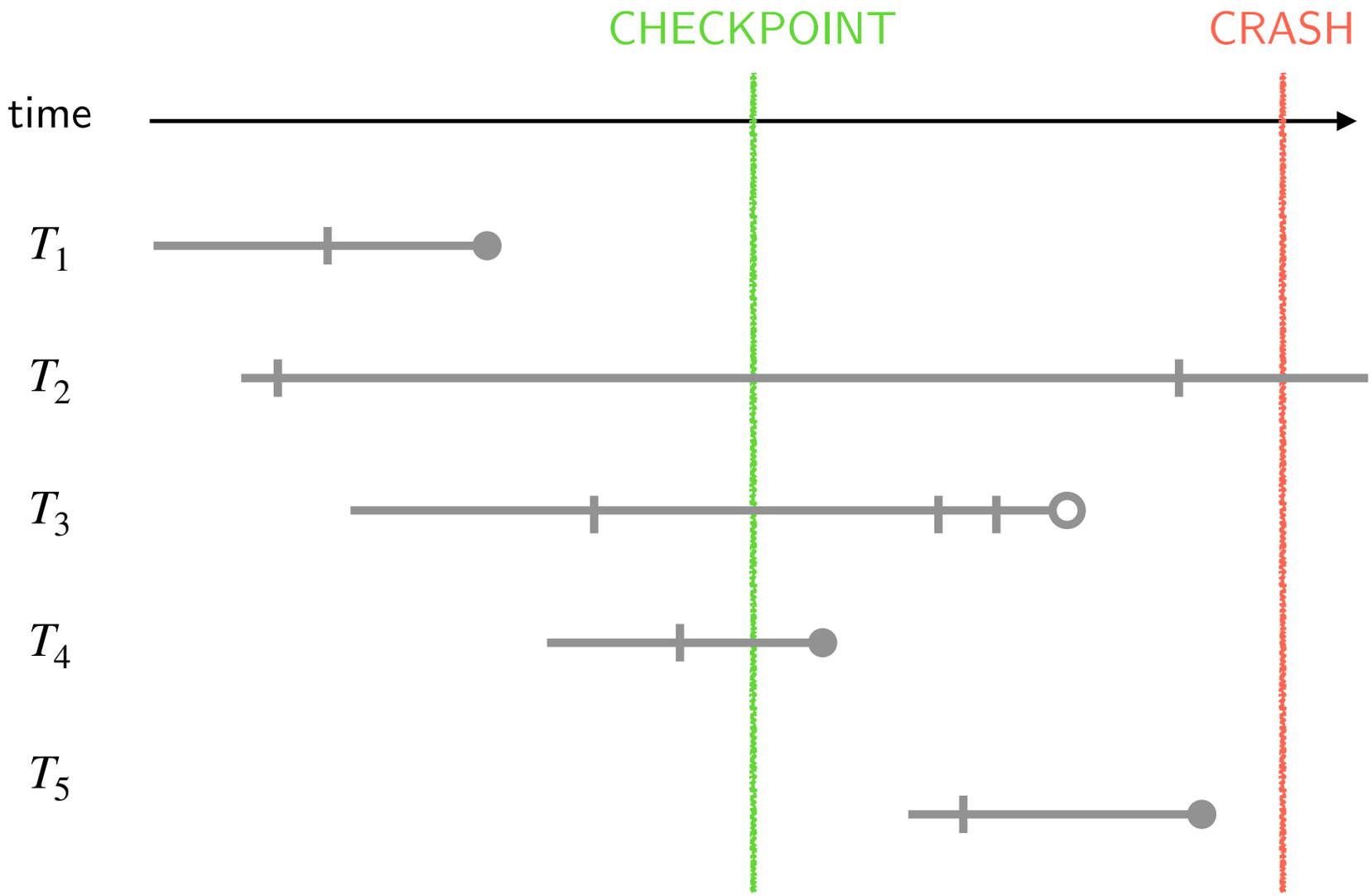


UNDO $= \{T_2, T_3, T_5\}$

REDO $= \{T_4\}$

Costruire UNDO e REDO

$B(T_1)$
$B(T_2)$
$U(T_2, O_1, B_1, A_1)$
$I(T_1, O_2, A_2)$
$B(T_3)$
$C(T_1)$
$B(T_4)$
$U(T_3, O_2, B_3, A_3)$
$U(T_4, O_3, B_4, A_4)$
$CK(T_2, T_3, T_4)$
$C(T_4)$
$B(T_5)$
$U(T_3, O_3, B_5, A_5)$
$U(T_5, O_4, B_6, A_6)$
$D(T_3, O_5, B_7)$
$A(T_3)$
$C(T_5)$
$I(T_2, O_6, A_8)$

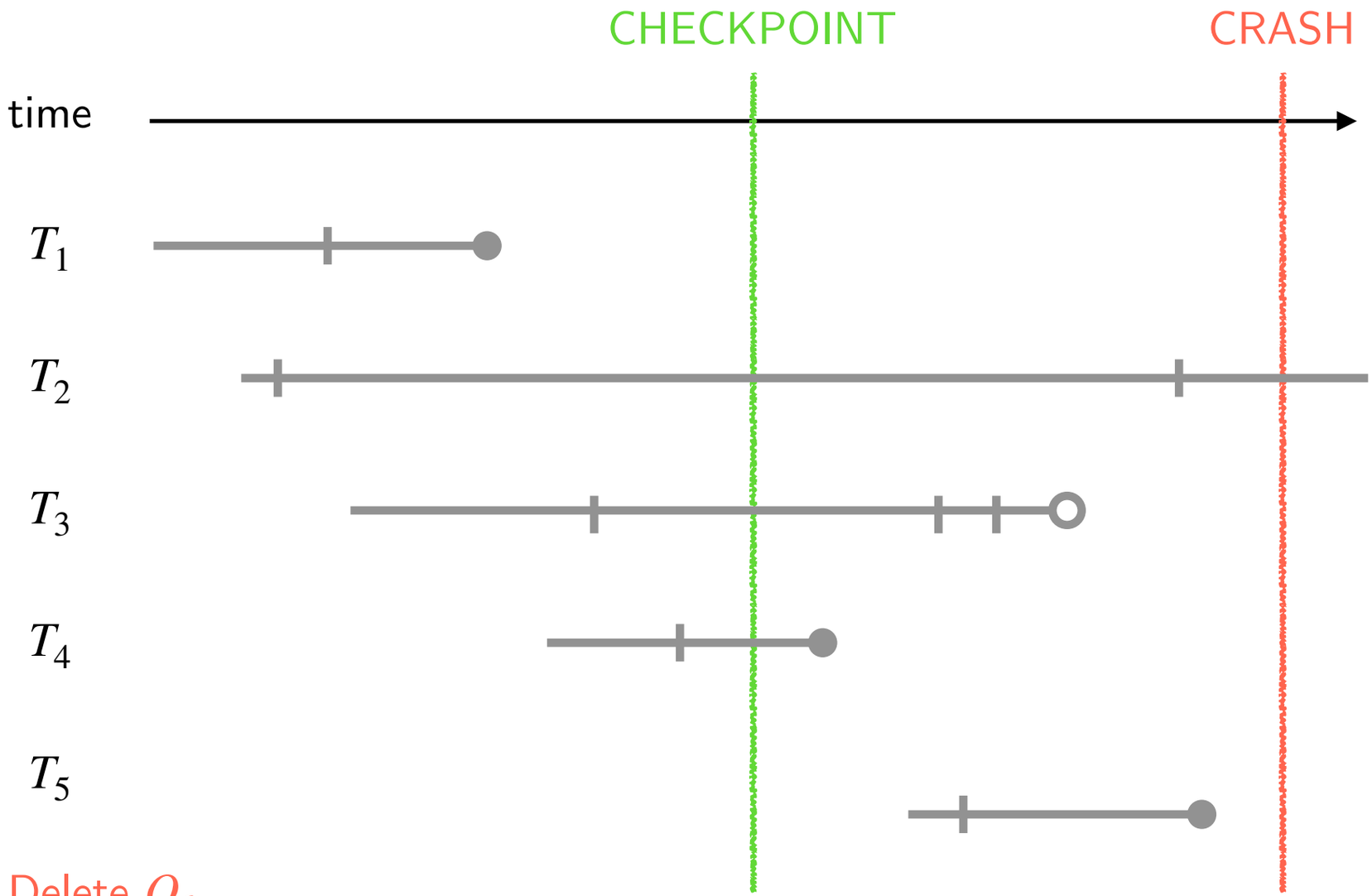


UNDO $= \{T_2, T_3\}$

REDO $= \{T_4, T_5\}$

Disfare gli UNDO a ritroso

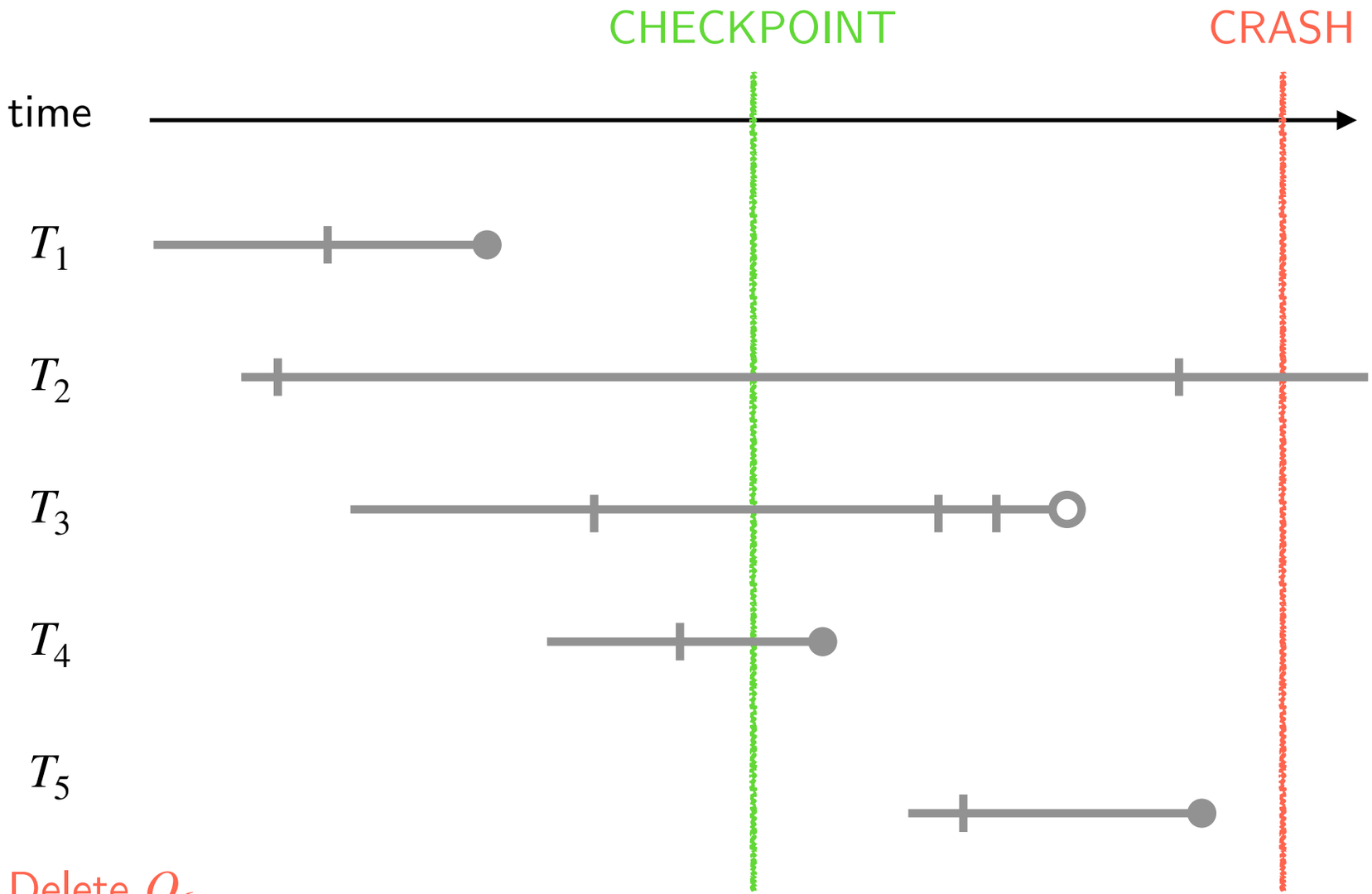
$B(T_1)$
$B(T_2)$
$U(T_2, O_1, B_1, A_1)$
$I(T_1, O_2, A_2)$
$B(T_3)$
$C(T_1)$
$B(T_4)$
$U(T_3, O_2, B_3, A_3)$
$U(T_4, O_3, B_4, A_4)$
$CK(T_2, T_3, T_4)$
$C(T_4)$
$B(T_5)$
$U(T_3, O_3, B_5, A_5)$
$U(T_5, O_4, B_6, A_6)$
$D(T_3, O_5, B_7)$
$A(T_3)$
$C(T_5)$
$I(T_2, O_6, A_8)$



UNDO $= \{T_2, T_3\}$

Disfare gli UNDO a ritroso

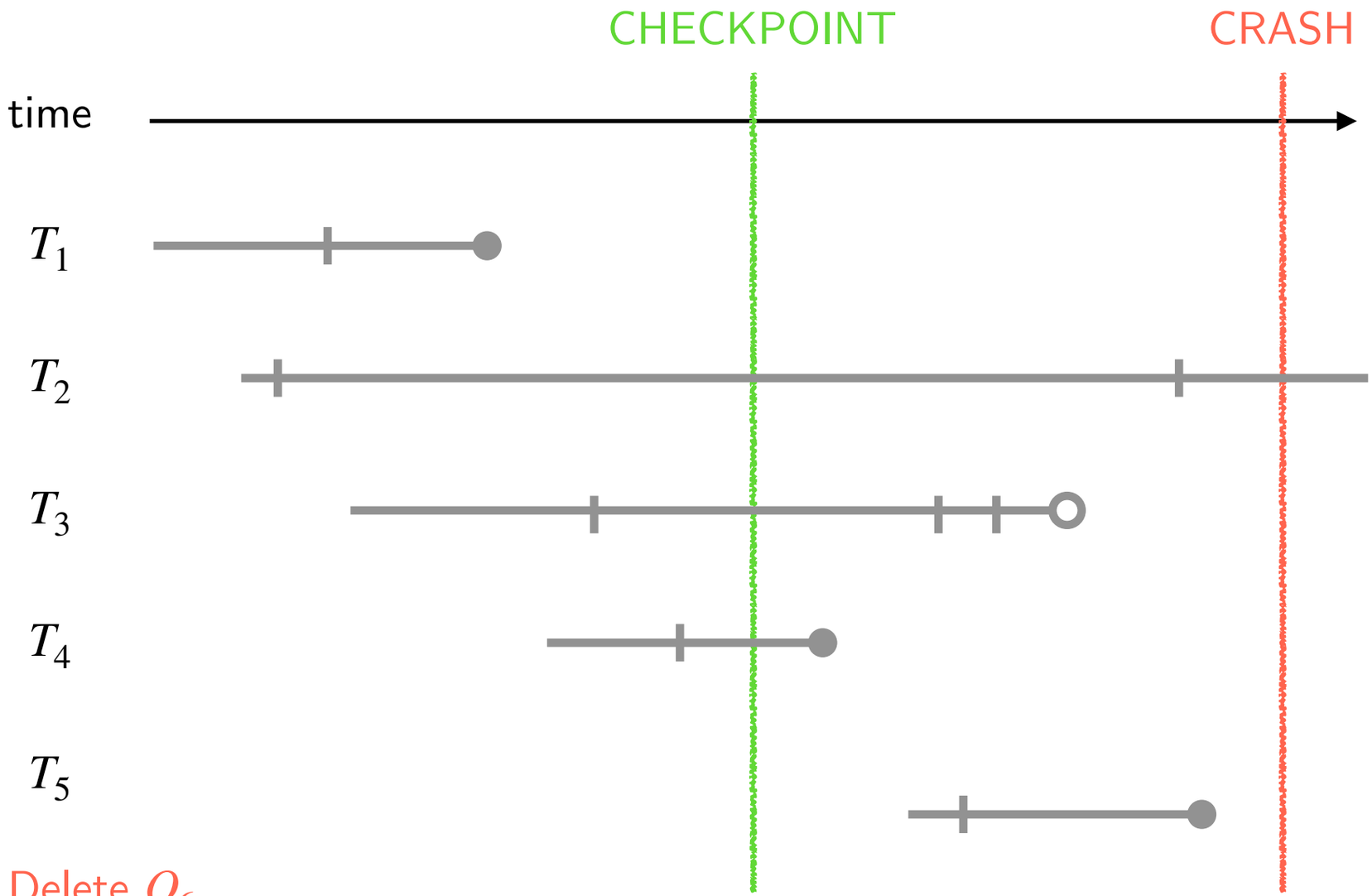
$B(T_1)$
$B(T_2)$
$U(T_2, O_1, B_1, A_1)$
$I(T_1, O_2, A_2)$
$B(T_3)$
$C(T_1)$
$B(T_4)$
$U(T_3, O_2, B_3, A_3)$
$U(T_4, O_3, B_4, A_4)$
$CK(T_2, T_3, T_4)$
$C(T_4)$
$B(T_5)$
$U(T_3, O_3, B_5, A_5)$
$U(T_5, O_4, B_6, A_6)$
$D(T_3, O_5, B_7)$ ← Delete O_6 / Reinsert B_7 in O_5
$A(T_3)$
$C(T_5)$
$I(T_2, O_6, A_8)$ ←



UNDO $= \{T_2, T_3\}$

Disfare gli UNDO a ritroso

$B(T_1)$
$B(T_2)$
$U(T_2, O_1, B_1, A_1)$
$I(T_1, O_2, A_2)$
$B(T_3)$
$C(T_1)$
$B(T_4)$
$U(T_3, O_2, B_3, A_3)$
$U(T_4, O_3, B_4, A_4)$
$CK(T_2, T_3, T_4)$
$C(T_4)$
$B(T_5)$
$U(T_3, O_3, B_5, A_5)$ ←
$U(T_5, O_4, B_6, A_6)$
$D(T_3, O_5, B_7)$ ←
$A(T_3)$
$C(T_5)$
$I(T_2, O_6, A_8)$ ←

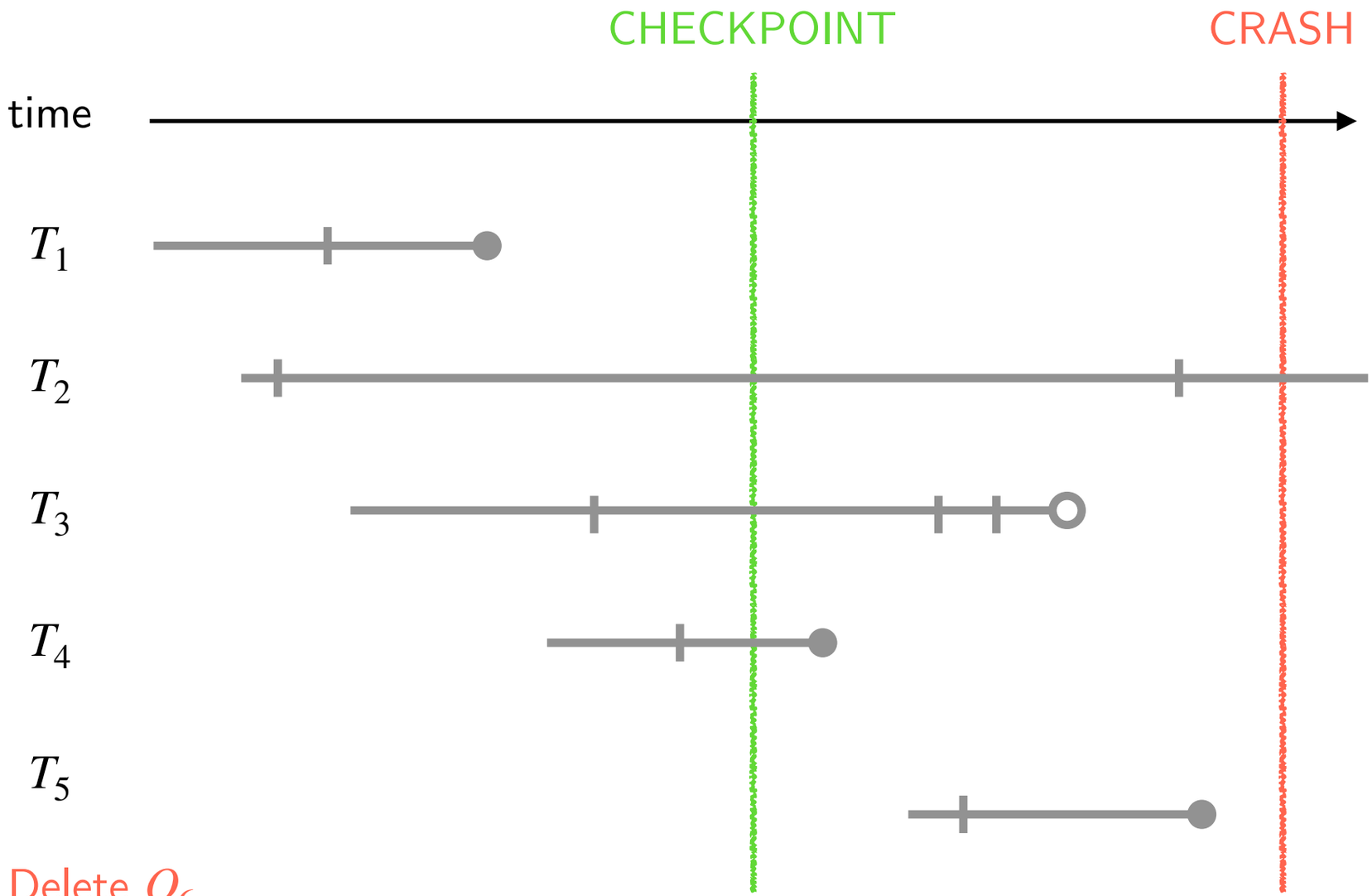


Delete O_6
Reinsert B_7 in O_5
$O_3 = B_5$

UNDO $= \{T_2, T_3\}$

Disfare gli UNDO a ritroso

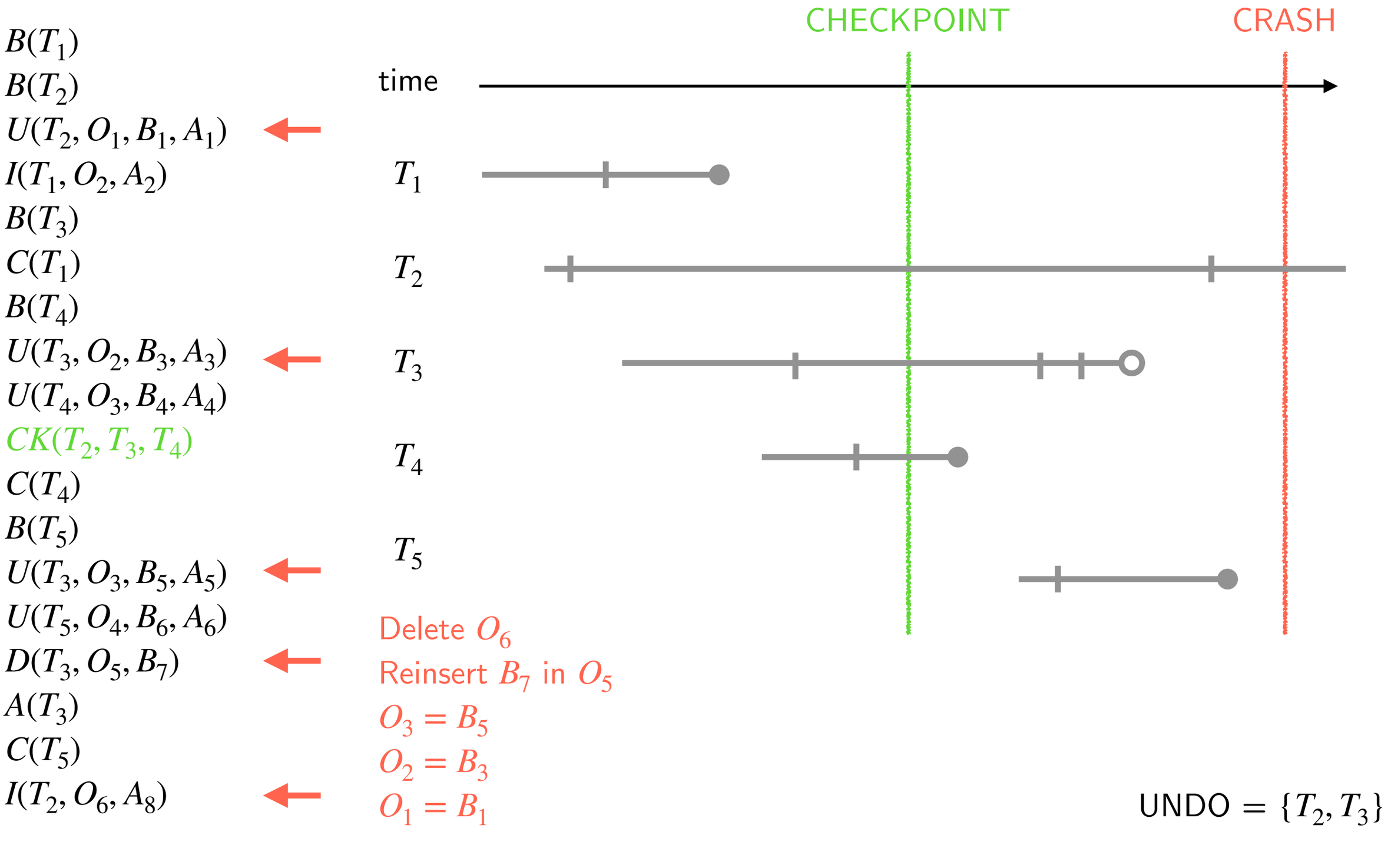
$B(T_1)$
$B(T_2)$
$U(T_2, O_1, B_1, A_1)$
$I(T_1, O_2, A_2)$
$B(T_3)$
$C(T_1)$
$B(T_4)$
$U(T_3, O_2, B_3, A_3)$ ←
$U(T_4, O_3, B_4, A_4)$
$CK(T_2, T_3, T_4)$
$C(T_4)$
$B(T_5)$
$U(T_3, O_3, B_5, A_5)$ ←
$U(T_5, O_4, B_6, A_6)$
$D(T_3, O_5, B_7)$ ←
$A(T_3)$
$C(T_5)$
$I(T_2, O_6, A_8)$ ←



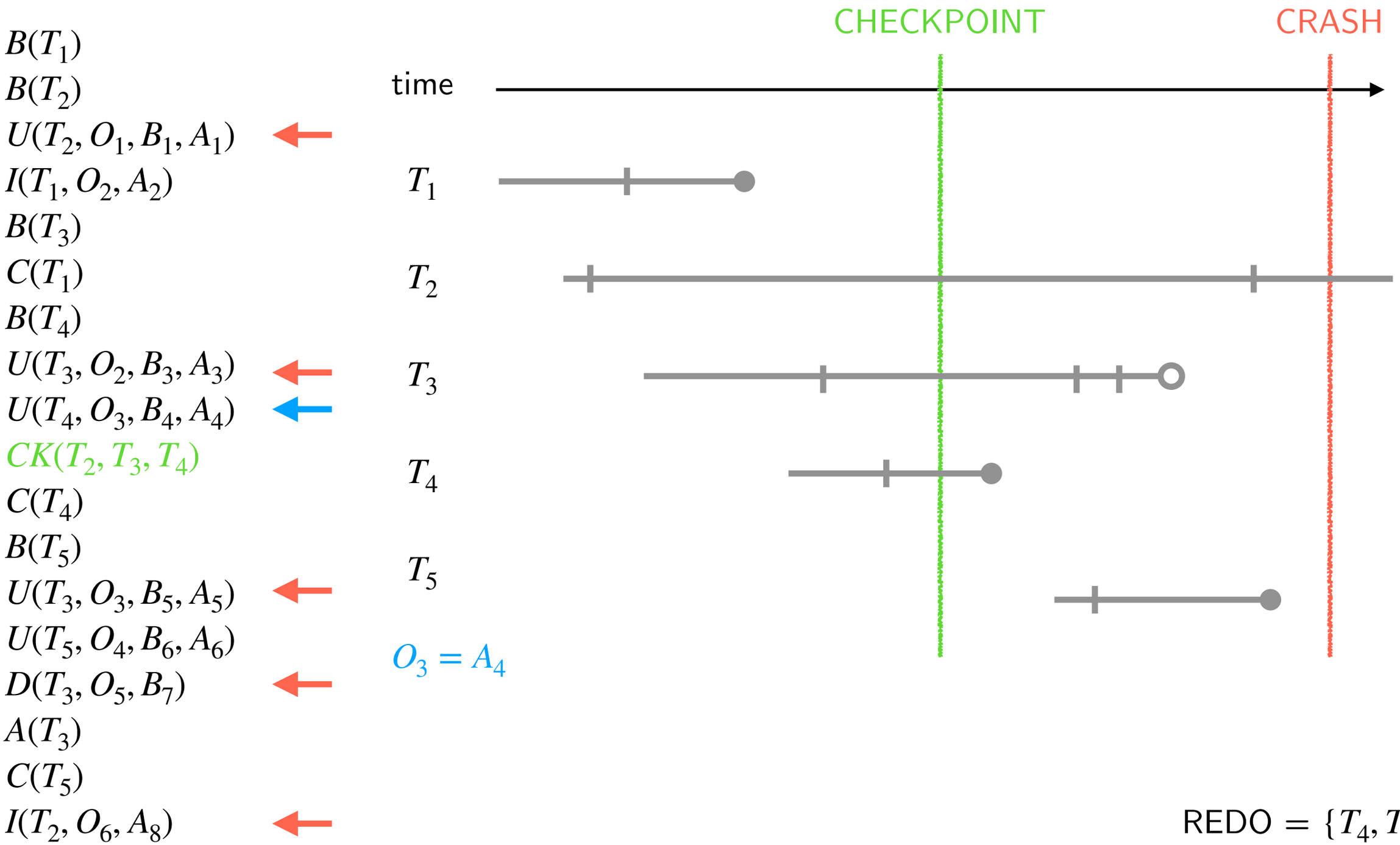
Delete O_6
Reinsert B_7 in O_5
$O_3 = B_5$
$O_2 = B_3$

UNDO $= \{T_2, T_3\}$

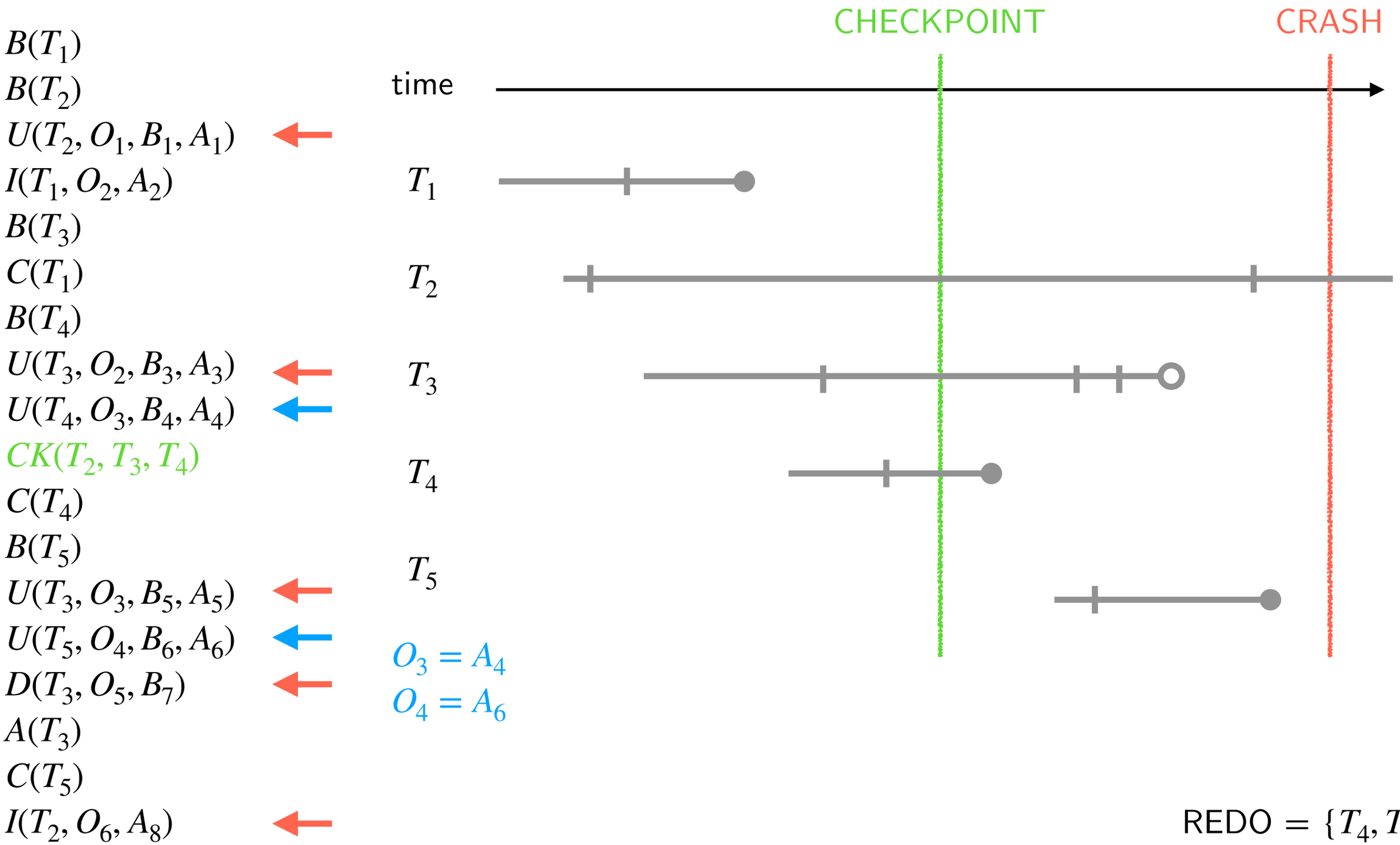
Disfare gli UNDO a ritroso



Rifare i REDO in avanti



Rifare i REDO in avanti



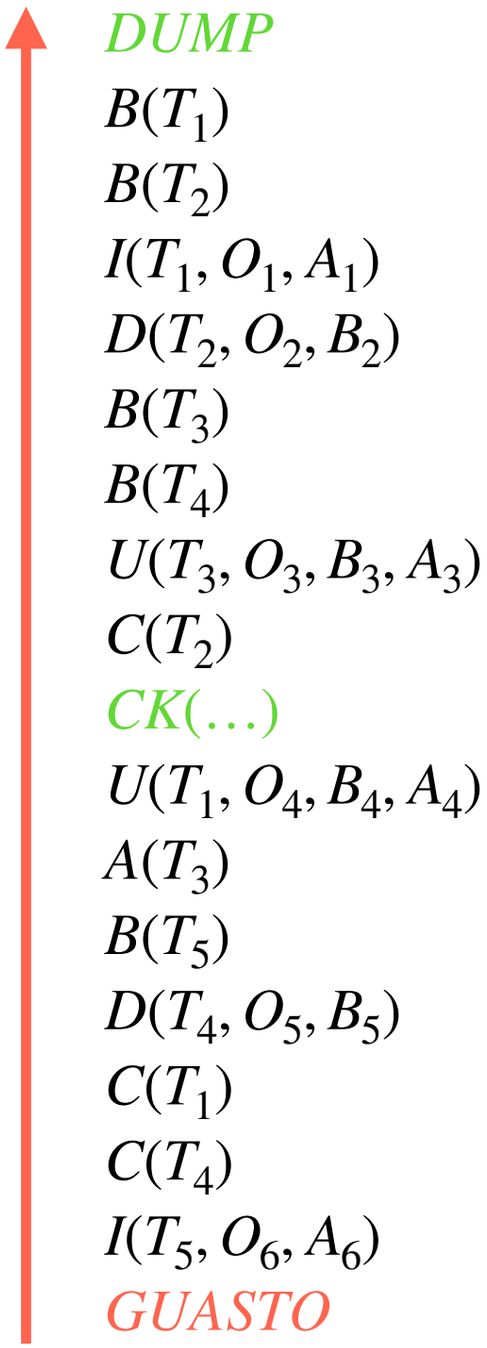
Ripresa a freddo

- Ci si riporta al **record di dump** più recente nel log e si ripristina la parte di dati deteriorata
- Si eseguono le operazioni registrate sul log sulla parte deteriorata fino all'istante del guasto
- Si esegue una ripresa a caldo

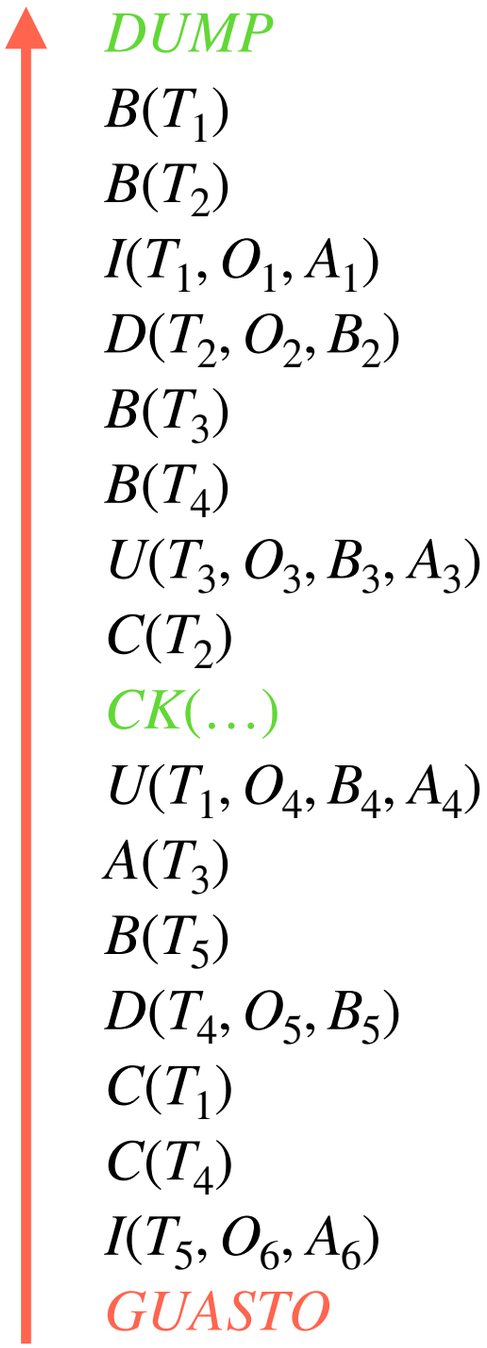
Esempio

DUMP
$B(T_1)$
$B(T_2)$
$I(T_1, O_1, A_1)$
$D(T_2, O_2, B_2)$
$B(T_3)$
$B(T_4)$
$U(T_3, O_3, B_3, A_3)$
$C(T_2)$
CK(...)
$U(T_1, O_4, B_4, A_4)$
$A(T_3)$
$B(T_5)$
$D(T_4, O_5, B_5)$
$C(T_1)$
$C(T_4)$
$I(T_5, O_6, A_6)$
GUASTO

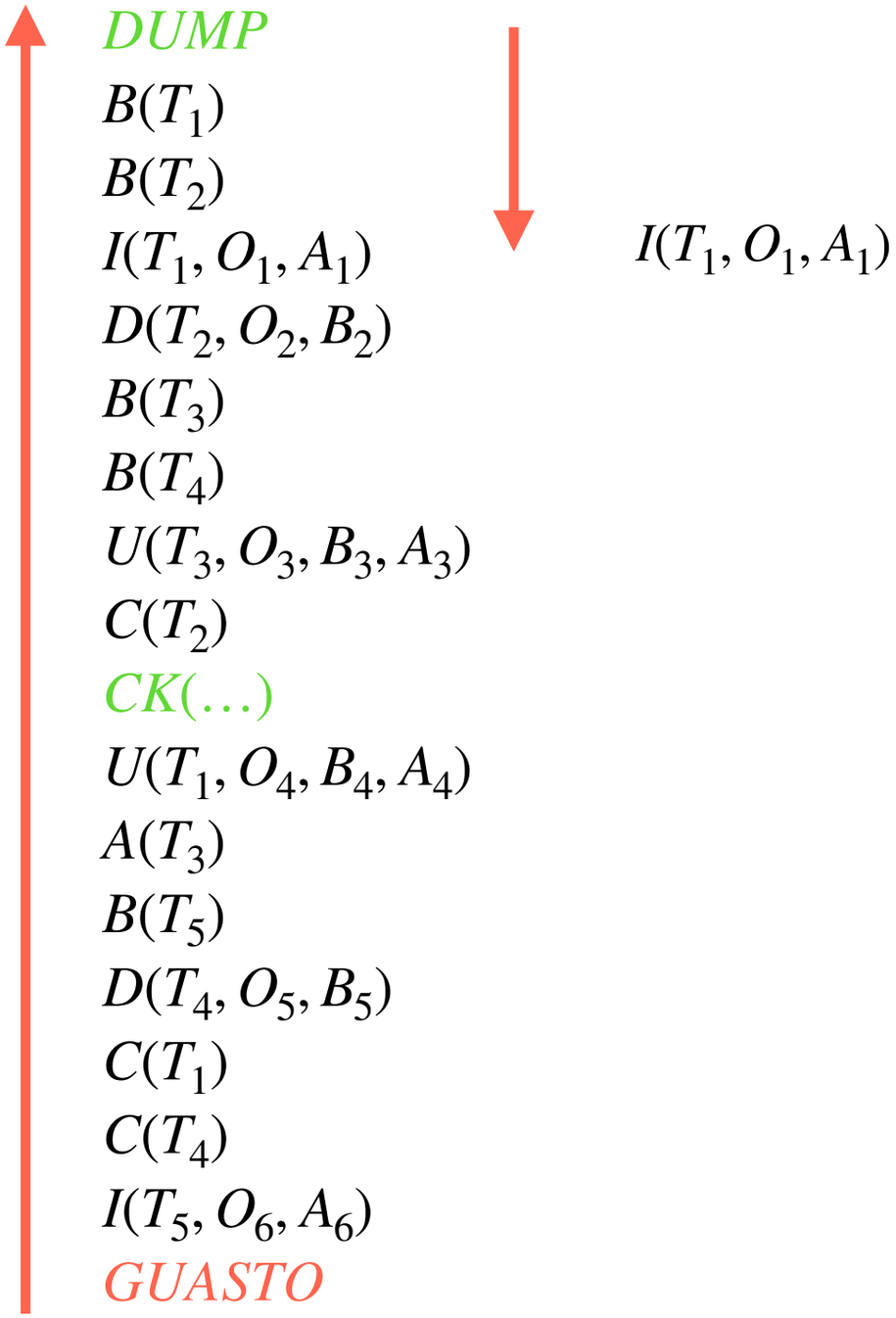
Esempio


DUMP
$B(T_1)$
$B(T_2)$
$I(T_1, O_1, A_1)$
$D(T_2, O_2, B_2)$
$B(T_3)$
$B(T_4)$
$U(T_3, O_3, B_3, A_3)$
$C(T_2)$
CK(...)
$U(T_1, O_4, B_4, A_4)$
$A(T_3)$
$B(T_5)$
$D(T_4, O_5, B_5)$
$C(T_1)$
$C(T_4)$
$I(T_5, O_6, A_6)$
GUASTO

Esempio

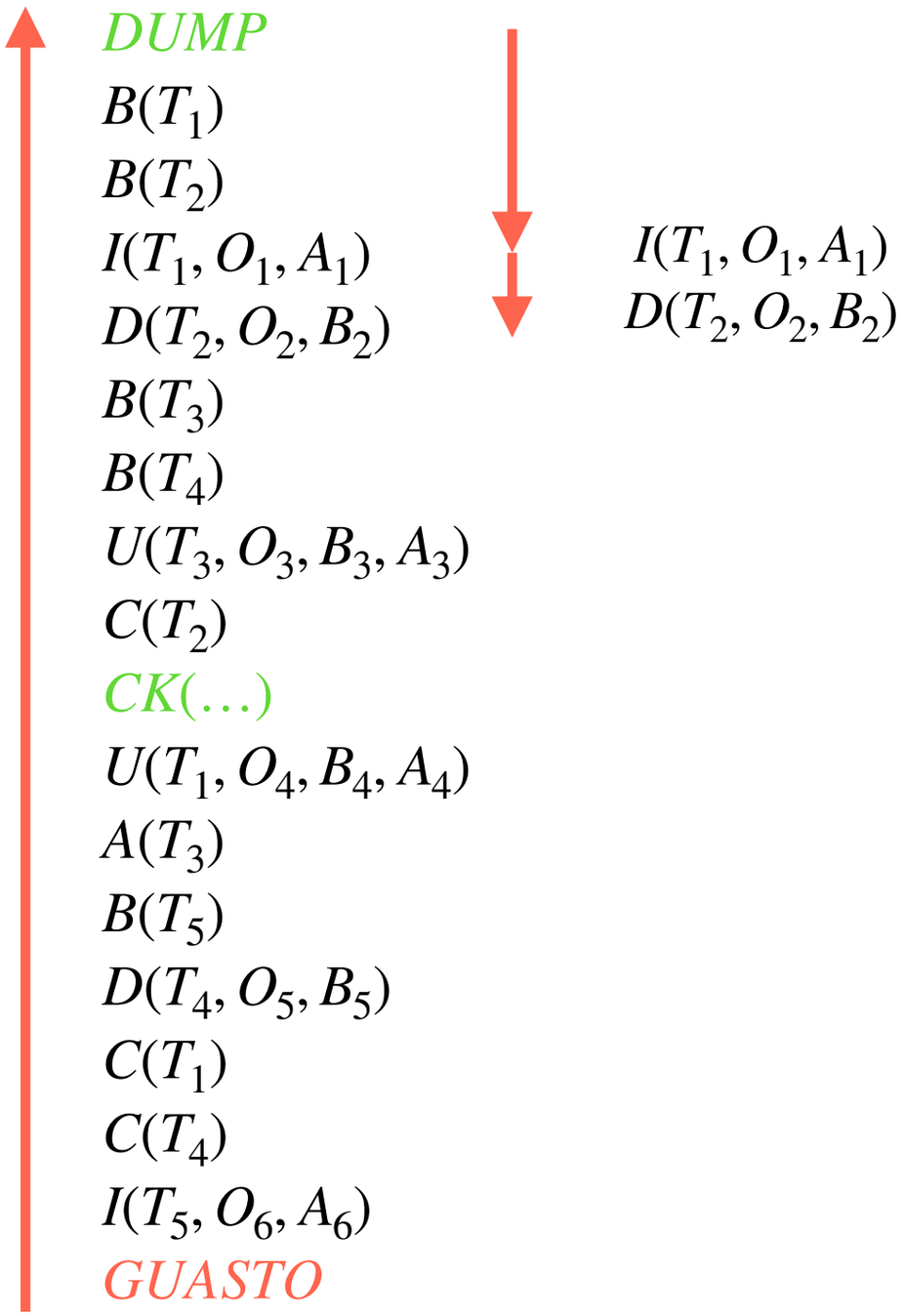

DUMP
$B(T_1)$
$B(T_2)$
$I(T_1, O_1, A_1)$
$D(T_2, O_2, B_2)$
$B(T_3)$
$B(T_4)$
$U(T_3, O_3, B_3, A_3)$
$C(T_2)$
CK(...)
$U(T_1, O_4, B_4, A_4)$
$A(T_3)$
$B(T_5)$
$D(T_4, O_5, B_5)$
$C(T_1)$
$C(T_4)$
$I(T_5, O_6, A_6)$
GUASTO

Esempio



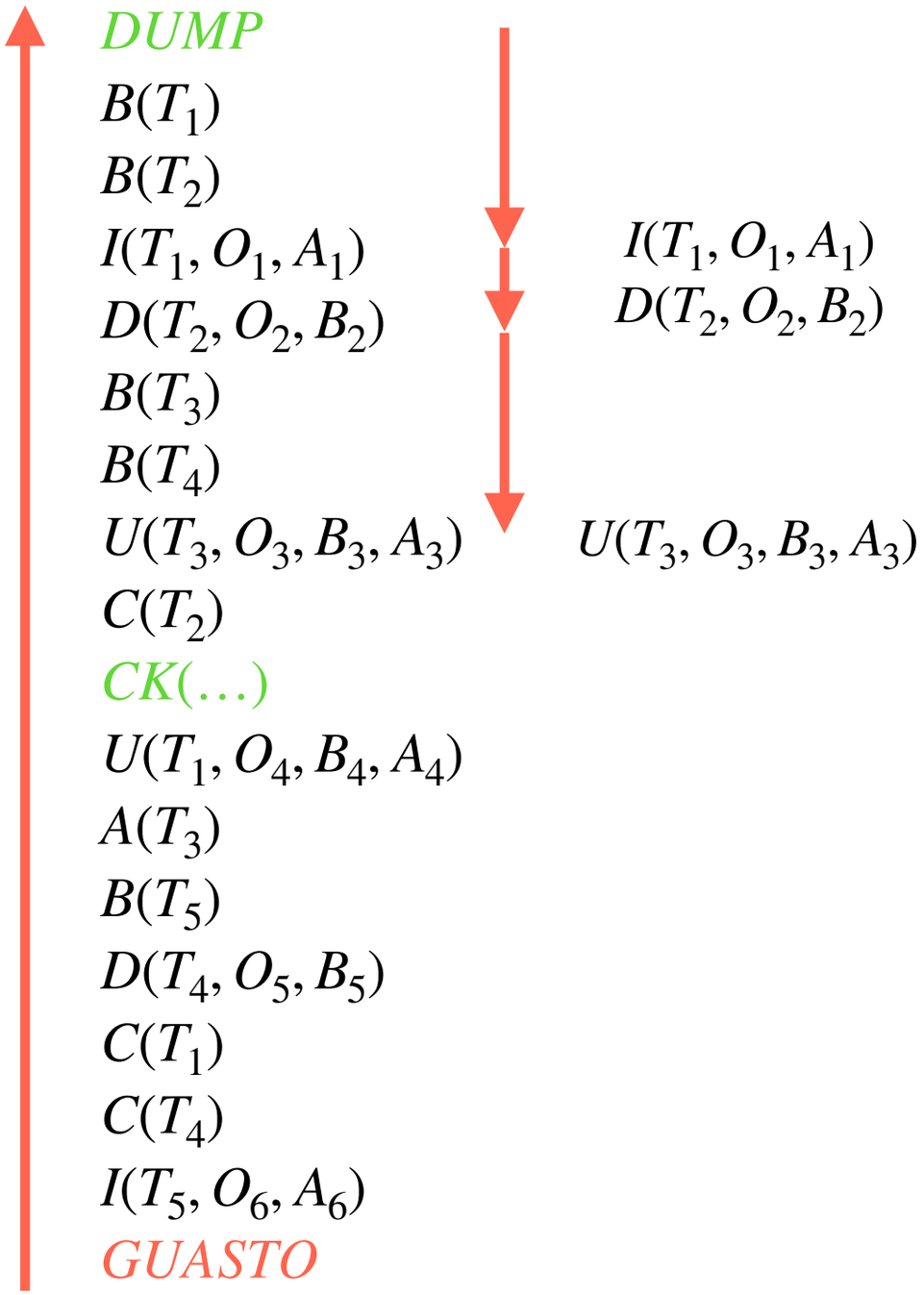
$I(T_1, O_1, A_1)$

Esempio


DUMP
$B(T_1)$
$B(T_2)$
$I(T_1, O_1, A_1)$
$D(T_2, O_2, B_2)$
$B(T_3)$
$B(T_4)$
$U(T_3, O_3, B_3, A_3)$
$C(T_2)$
CK(...)
$U(T_1, O_4, B_4, A_4)$
$A(T_3)$
$B(T_5)$
$D(T_4, O_5, B_5)$
$C(T_1)$
$C(T_4)$
$I(T_5, O_6, A_6)$
GUASTO

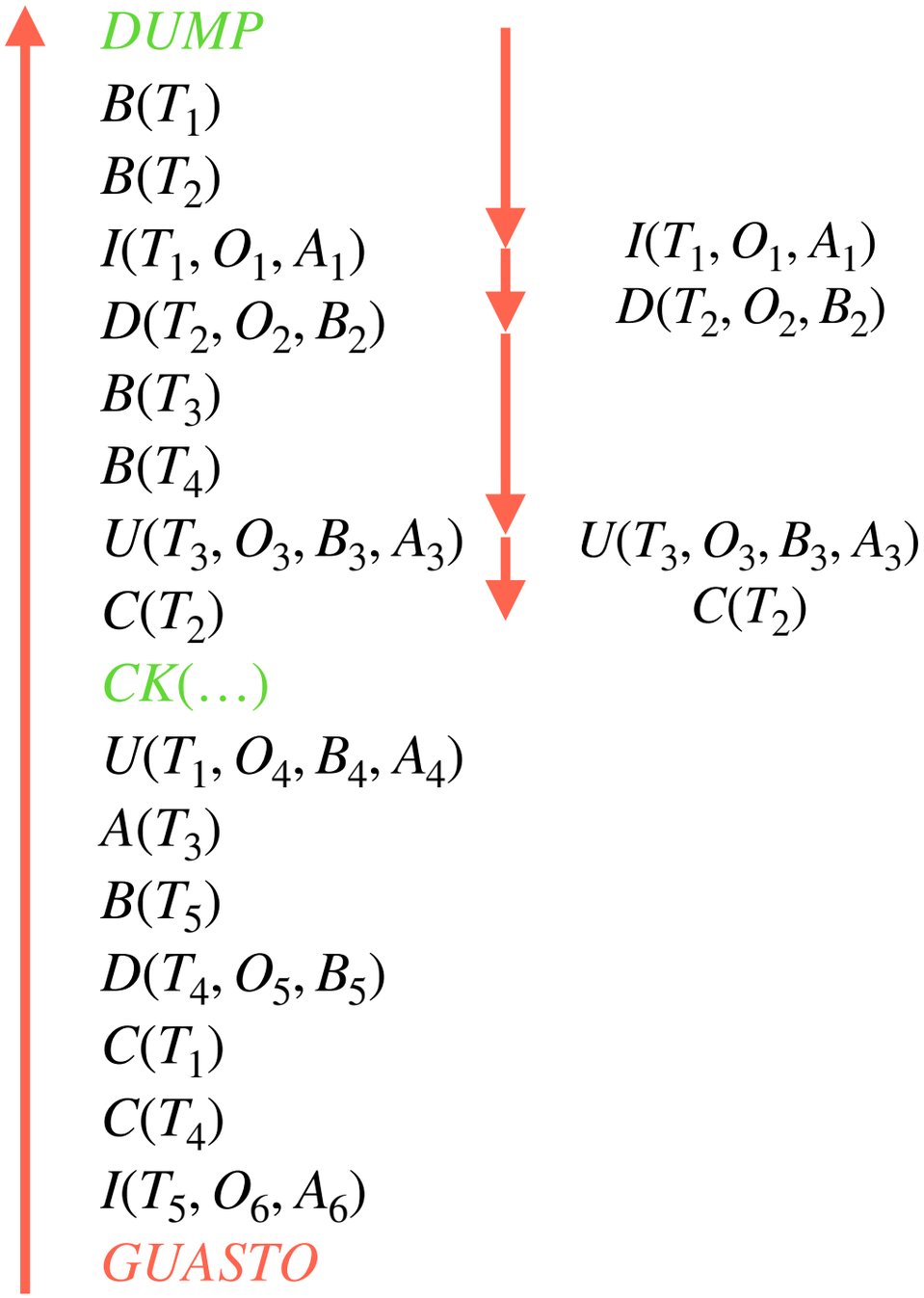
$I(T_1, O_1, A_1)$
$D(T_2, O_2, B_2)$

Esempio

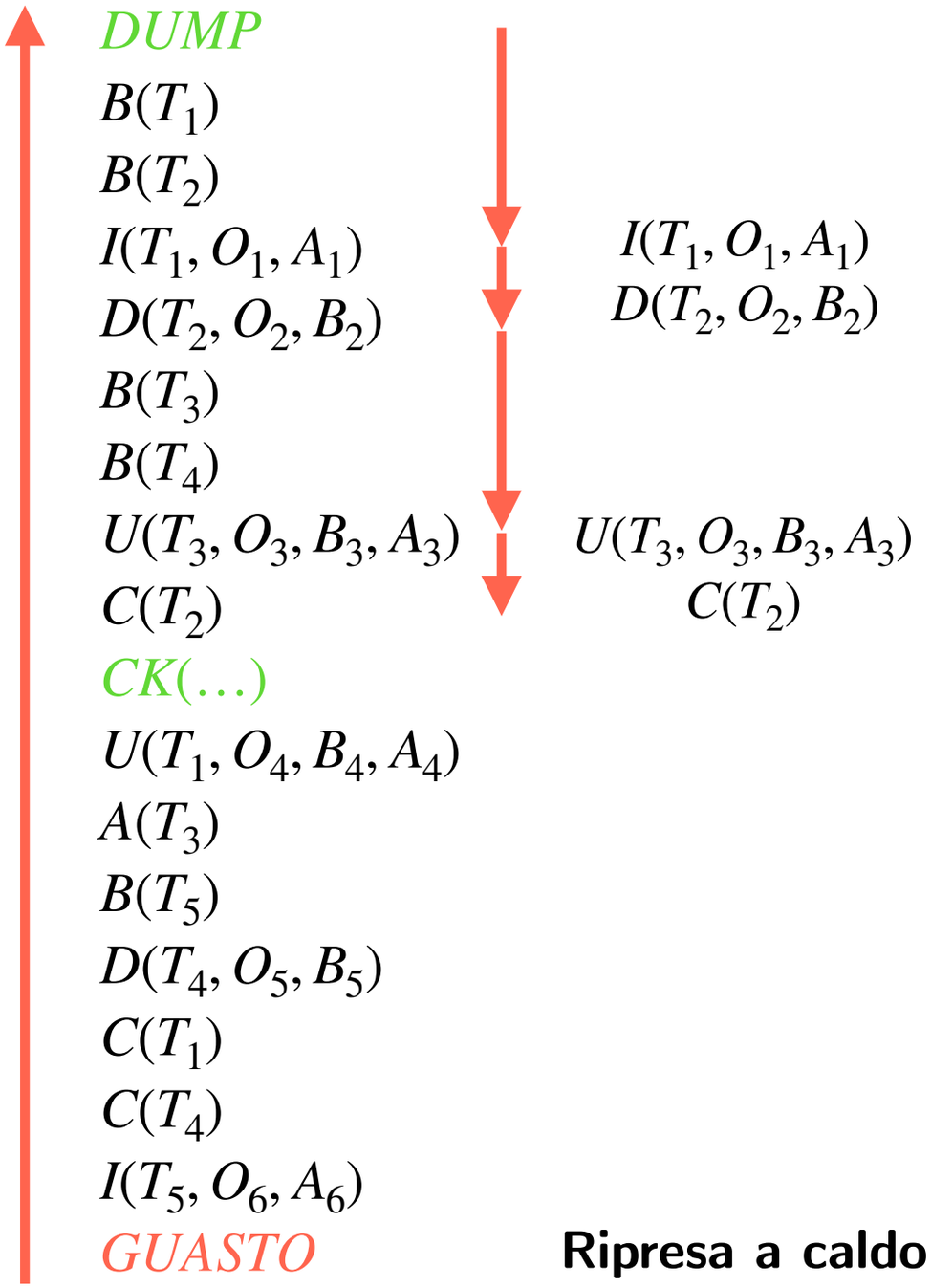

DUMP
$B(T_1)$
$B(T_2)$
$I(T_1, O_1, A_1)$
$D(T_2, O_2, B_2)$
$B(T_3)$
$B(T_4)$
$U(T_3, O_3, B_3, A_3)$
$C(T_2)$
CK(...)
$U(T_1, O_4, B_4, A_4)$
$A(T_3)$
$B(T_5)$
$D(T_4, O_5, B_5)$
$C(T_1)$
$C(T_4)$
$I(T_5, O_6, A_6)$
GUASTO

$I(T_1, O_1, A_1)$
$D(T_2, O_2, B_2)$
$U(T_3, O_3, B_3, A_3)$

Esempio


DUMP
$B(T_1)$
$B(T_2)$
$I(T_1, O_1, A_1)$
$D(T_2, O_2, B_2)$
$B(T_3)$
$B(T_4)$
$U(T_3, O_3, B_3, A_3)$
$C(T_2)$
CK(...)
$U(T_1, O_4, B_4, A_4)$
$A(T_3)$
$B(T_5)$
$D(T_4, O_5, B_5)$
$C(T_1)$
$C(T_4)$
$I(T_5, O_6, A_6)$
GUASTO
$I(T_1, O_1, A_1)$
$D(T_2, O_2, B_2)$
$U(T_3, O_3, B_3, A_3)$
$C(T_2)$

Esempio



Ripresa a caldo