Animace synchronizačních mechanizmů

PB173 Tématicky zaměřený vývoj aplikací v jazyce C/C++
Systémové programování – Linux

Roman Lacko

Fakulta informatiky
Masarykova univerzita
xlacko1@fi.muni.cz

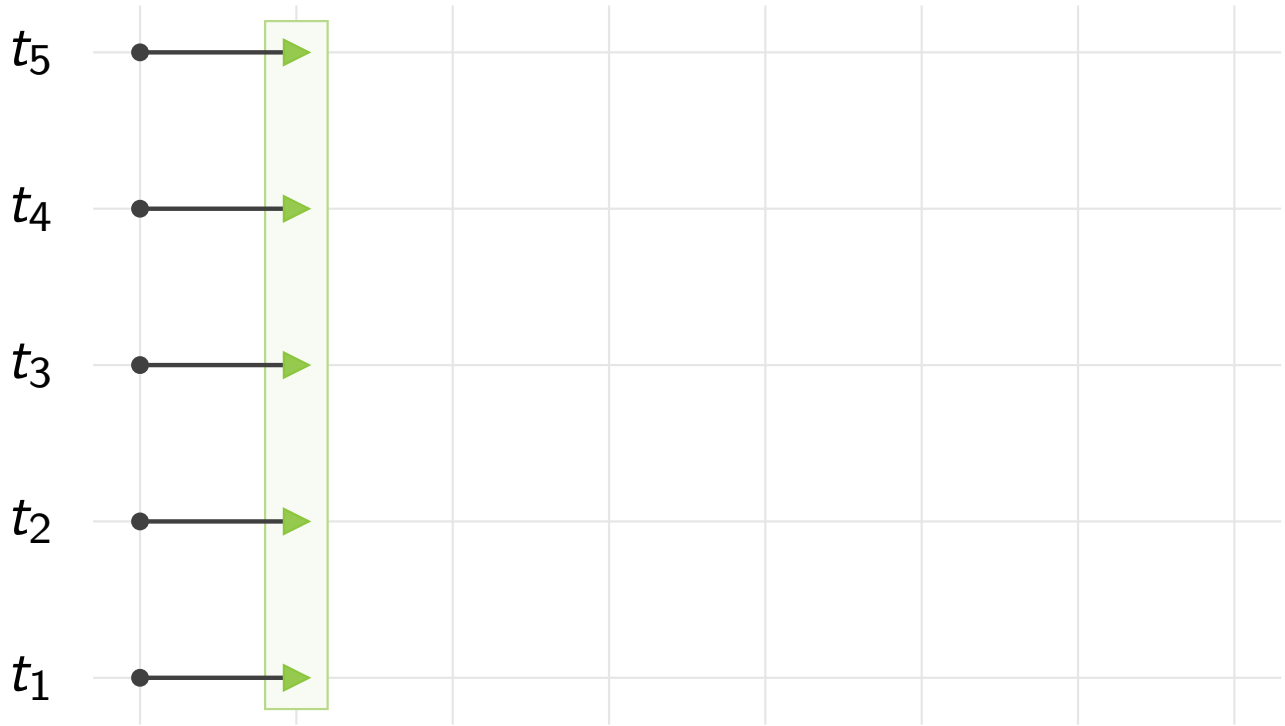
1. 11. 2021

Obsah

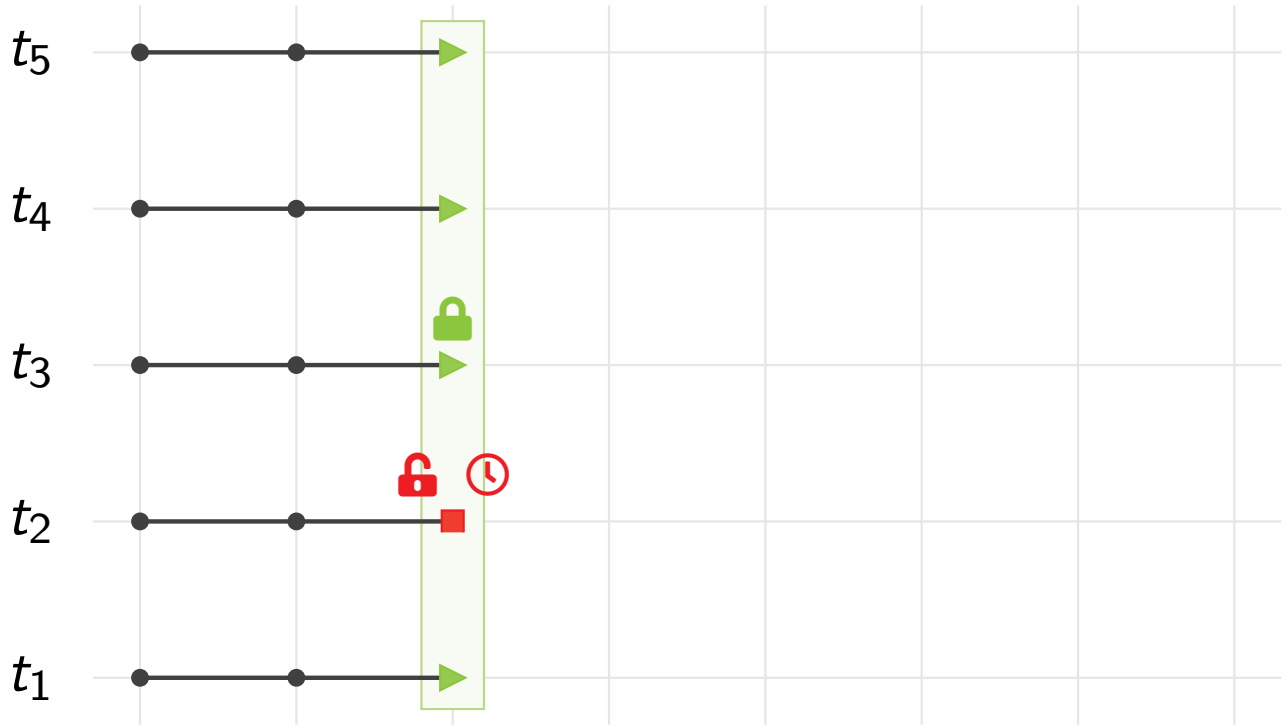
Mutex

```c
void *worker(void *arg)
{
    while (!/* finished */) {
        /* some time-consuming computation that produces a result */

        pthread_mutex_lock(&mutex);
        /* -- enter critical section -- */
        /* pass result to a consumer (terminal, pipe, ...) */
        /* -- leave critical section -- */
        pthread_mutex_unlock(&mutex);
    }

    // ...
}
```

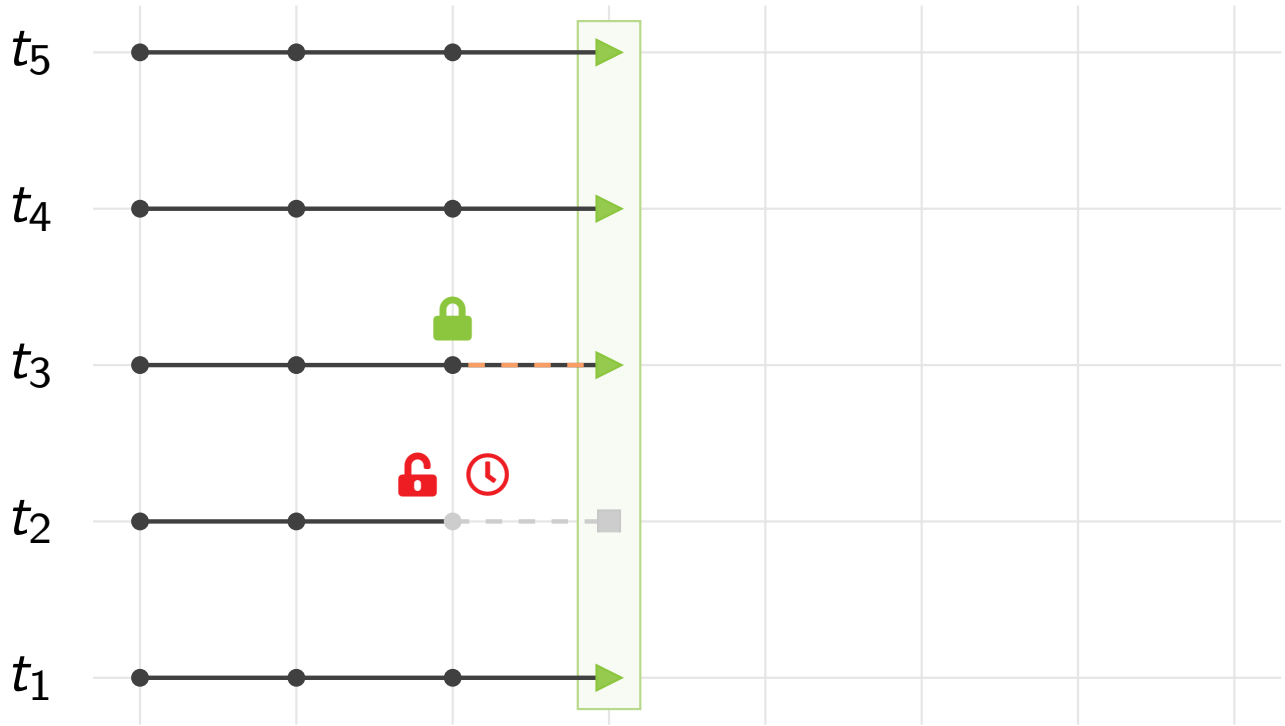
- t_5 běží
- t_4 běží
- t_3 běží
- t_2 běží
- t_1 běží



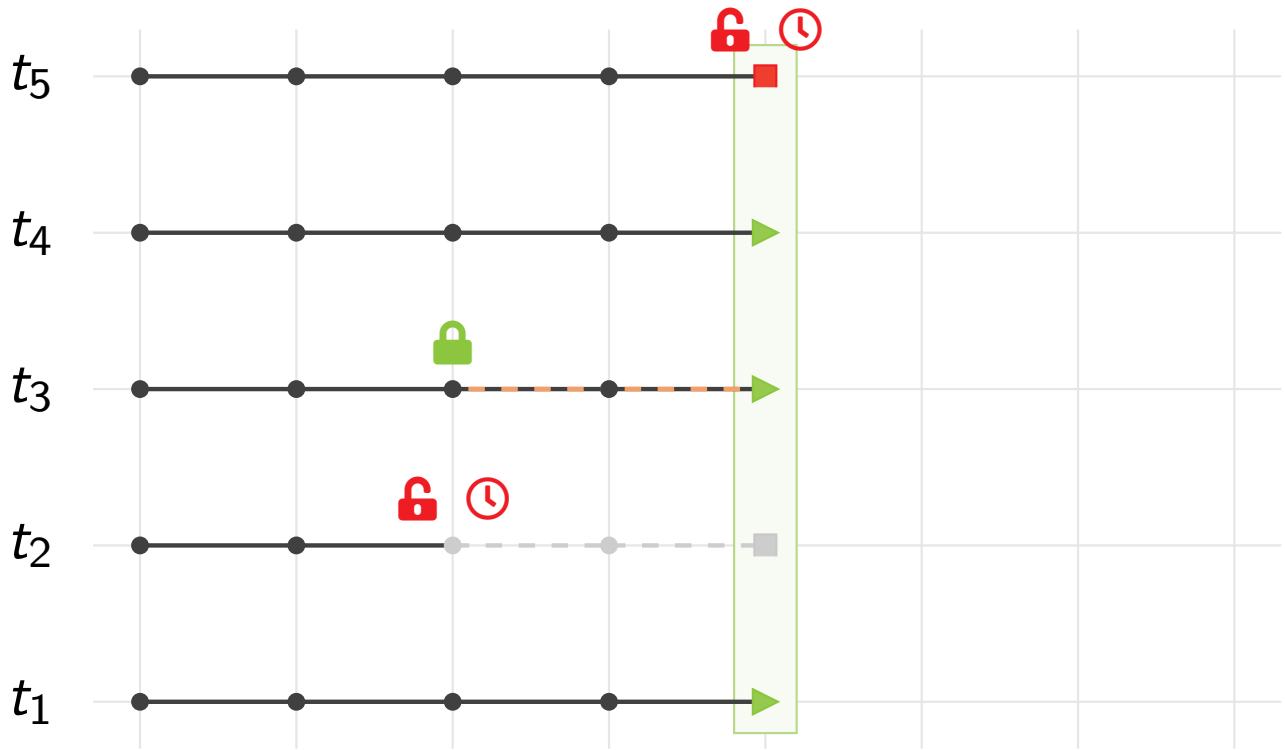
- t_5 běží
- t_4 běží
- t_3 běží
- t_2 běží
- t_1 běží



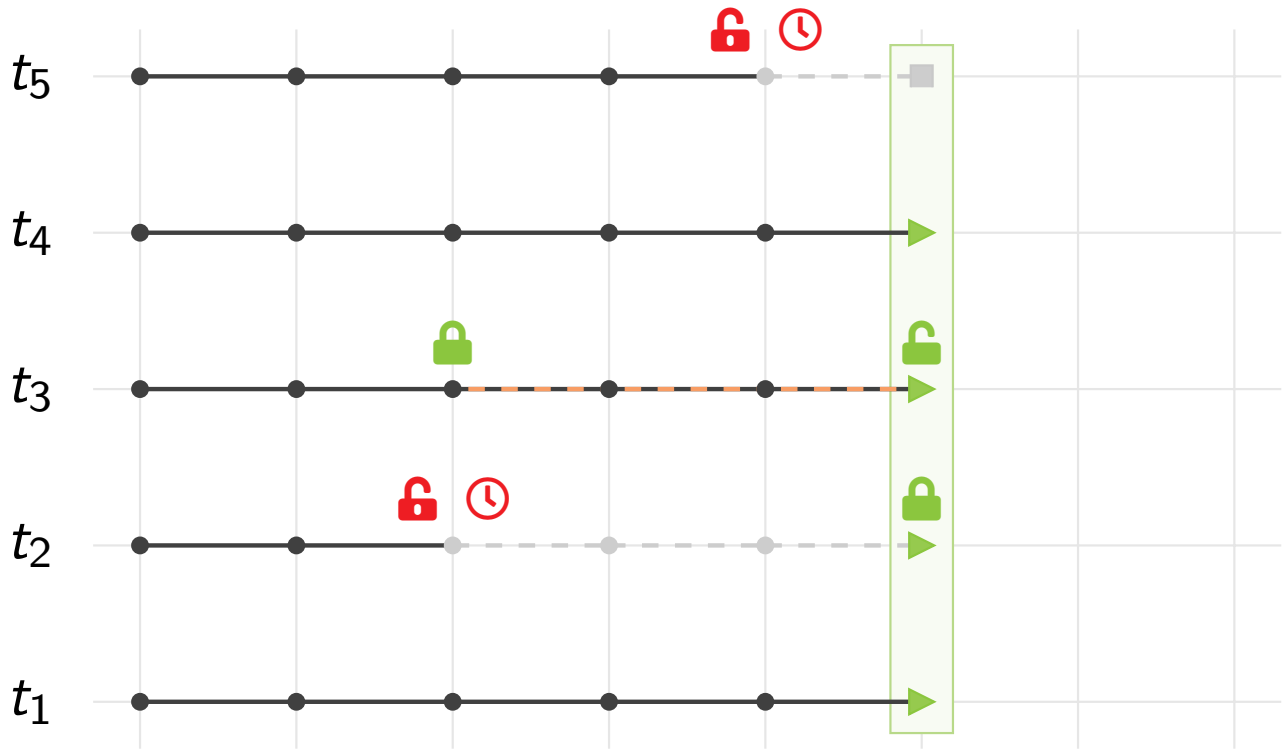
- t_5 běží
- t_4 běží
- t_3 `pthread_mutex_lock()`, vstupuje do kritické sekce
- t_2 `pthread_mutex_lock()`, uspí se
- t_1 běží



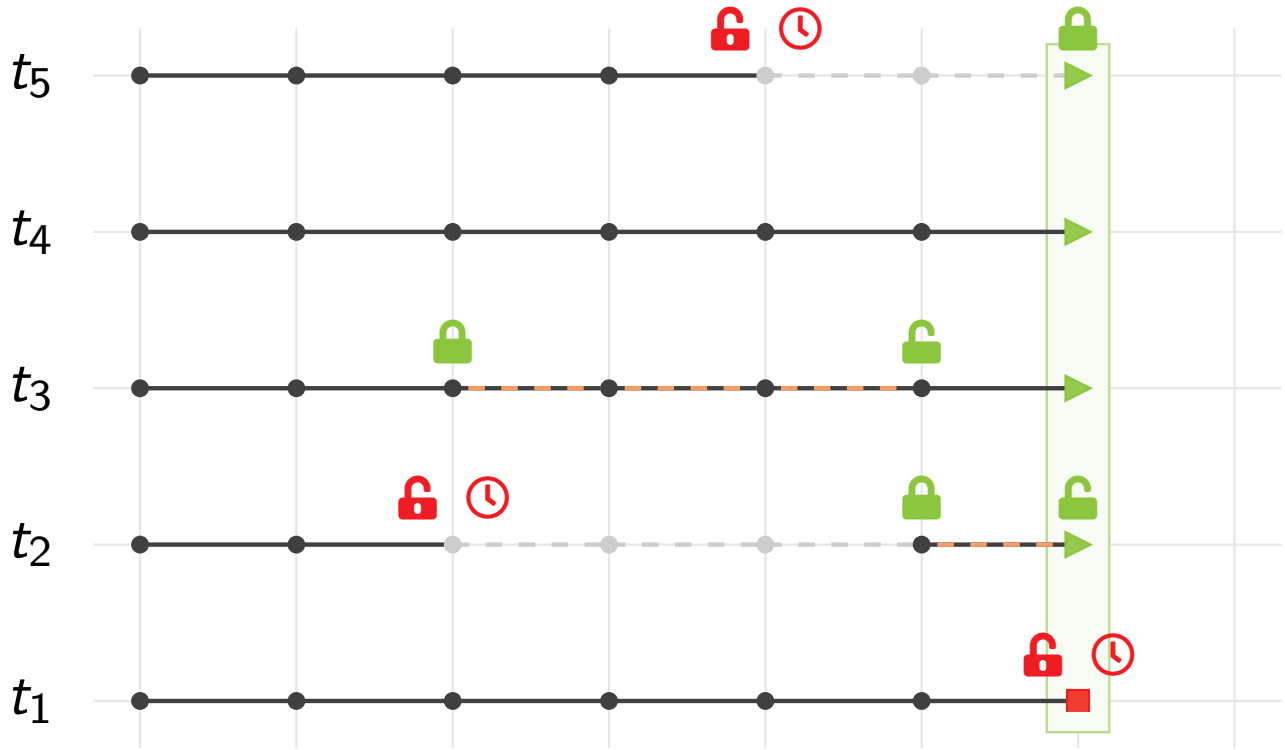
- t_5 běží
- t_4 běží
- t_3 běží v kritické sekci
- t_2 `pthread_mutex_lock()`, čeká
- t_1 běží



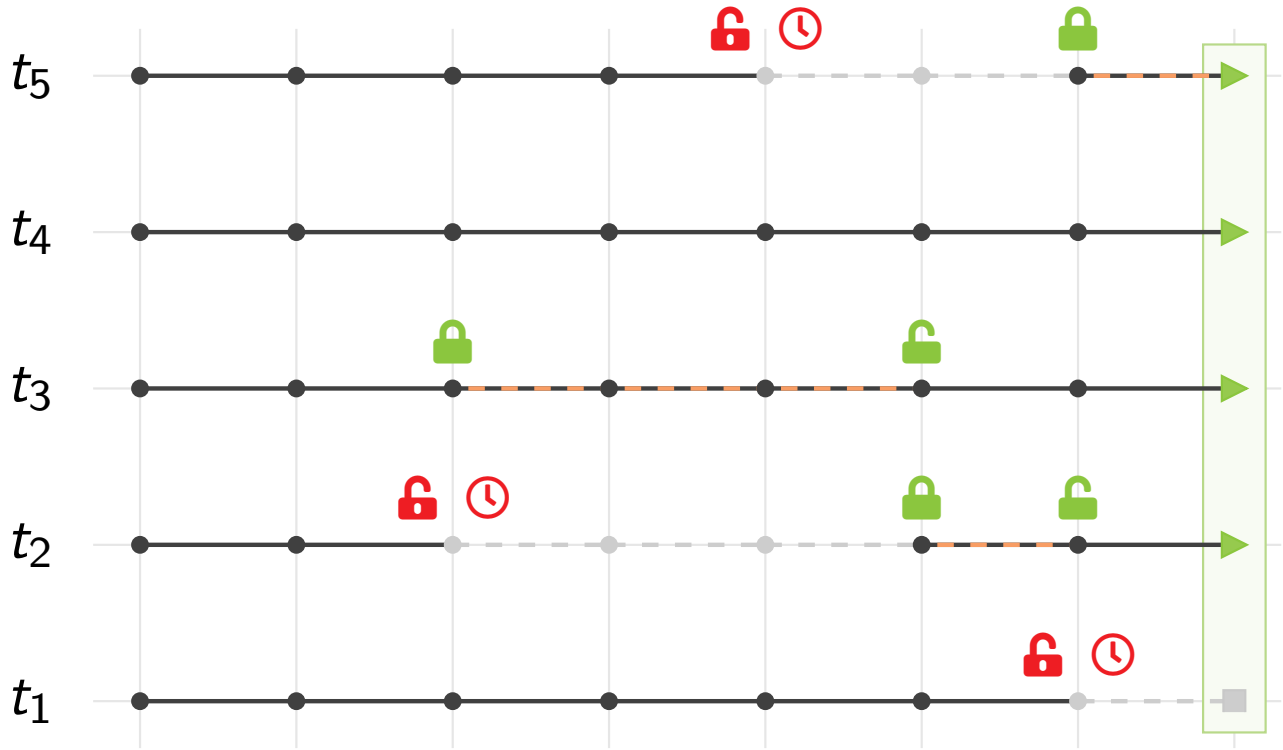
- t_5 `pthread_mutex_lock()`, uspí se
- t_4 běží
- t_3 běží v kritické sekci
- t_2 `pthread_mutex_lock()`, čeká
- t_1 běží



- t_5 `pthread_mutex_lock()`, čeká
- t_4 běží
- t_3 `pthread_mutex_unlock()`, opouští kritickou sekci
- t_2 `pthread_mutex_lock()`, probudí se a vstupuje do kritické sekce
- t_1 běží



- t_5 `pthread_mutex_lock()`, probudí se a vstupuje do kritické sekce
- t_4 běží
- t_3 běží
- t_2 `pthread_mutex_unlock()`, opouští kritickou sekci
- t_1 `pthread_mutex_lock()`, uspí se

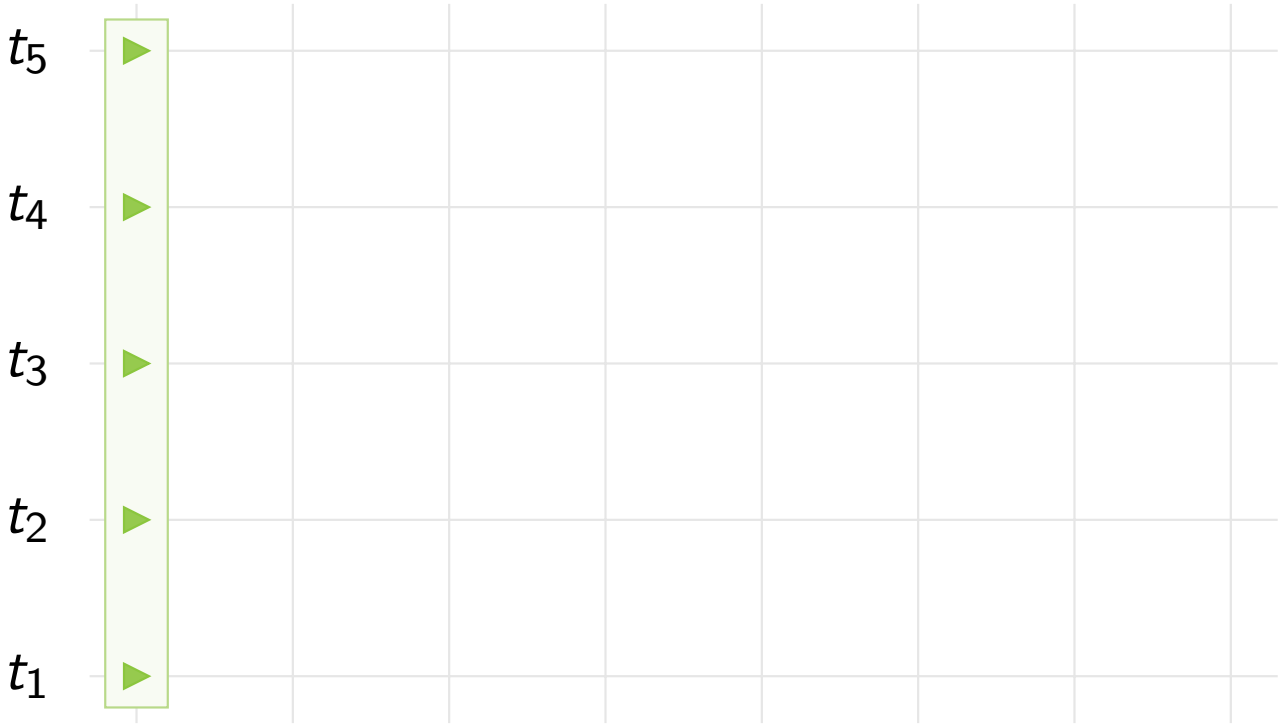


- t_5 běží v kritické sekci
- t_4 běží
- t_3 běží
- t_2 běží
- t_1 `pthread_mutex_lock()`, čeká

Spinlock

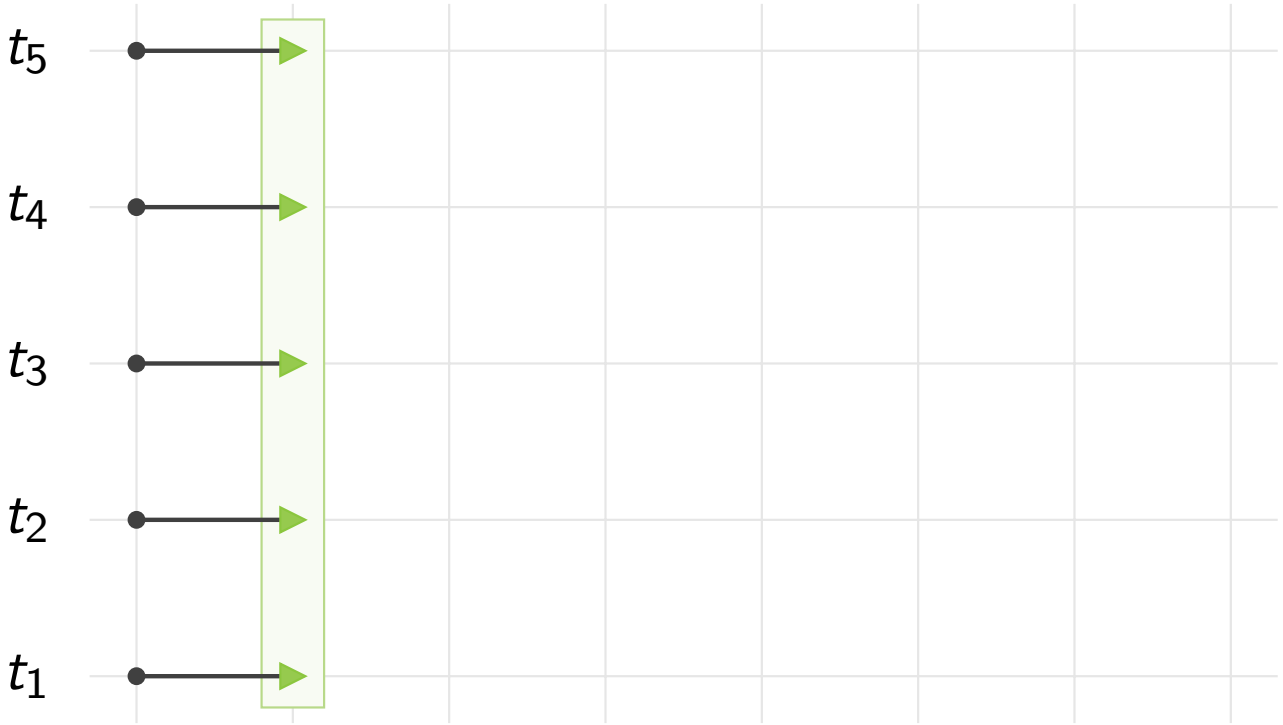
```c
void *worker(void *arg)
{
    while (!/* finished */) {
        /* some time-consuming computation that produces a result */

        pthread_spin_lock(&mutex);
        /* -- enter critical section -- */

        /* pass result to a consumer (terminal, pipe, ...) */

        /* -- leave critical section -- */
        pthread_spin_unlock(&mutex);
    }

    // ...
}
```

Spinlock



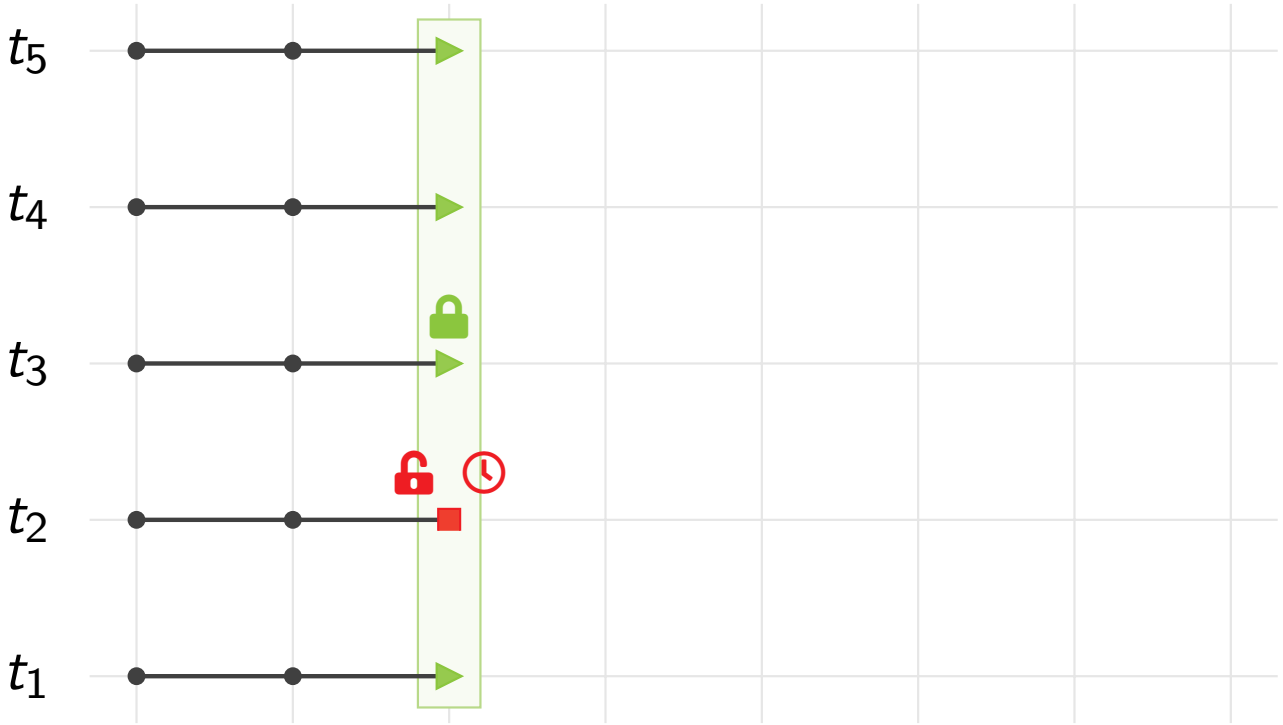
- t_5 běží
- t_4 běží
- t_3 běží
- t_2 běží
- t_1 běží

Spinlock



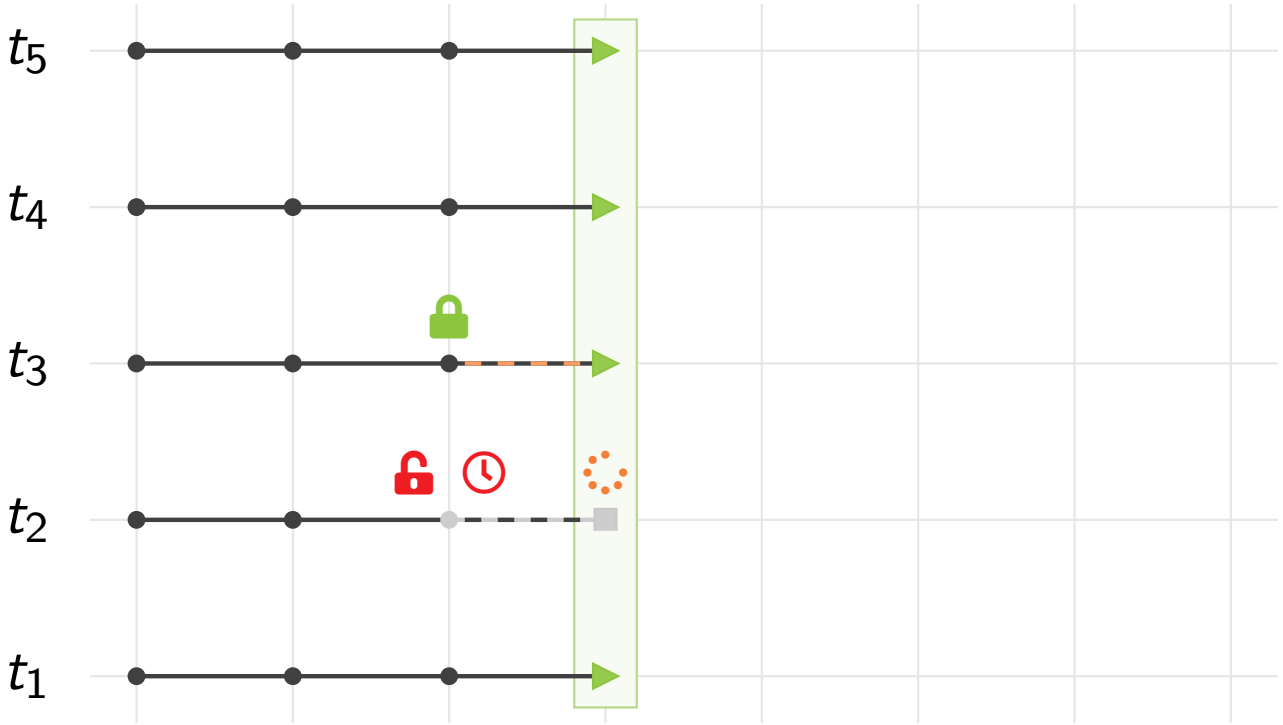
- t_5 běží
- t_4 běží
- t_3 běží
- t_2 běží
- t_1 běží

Spinlock



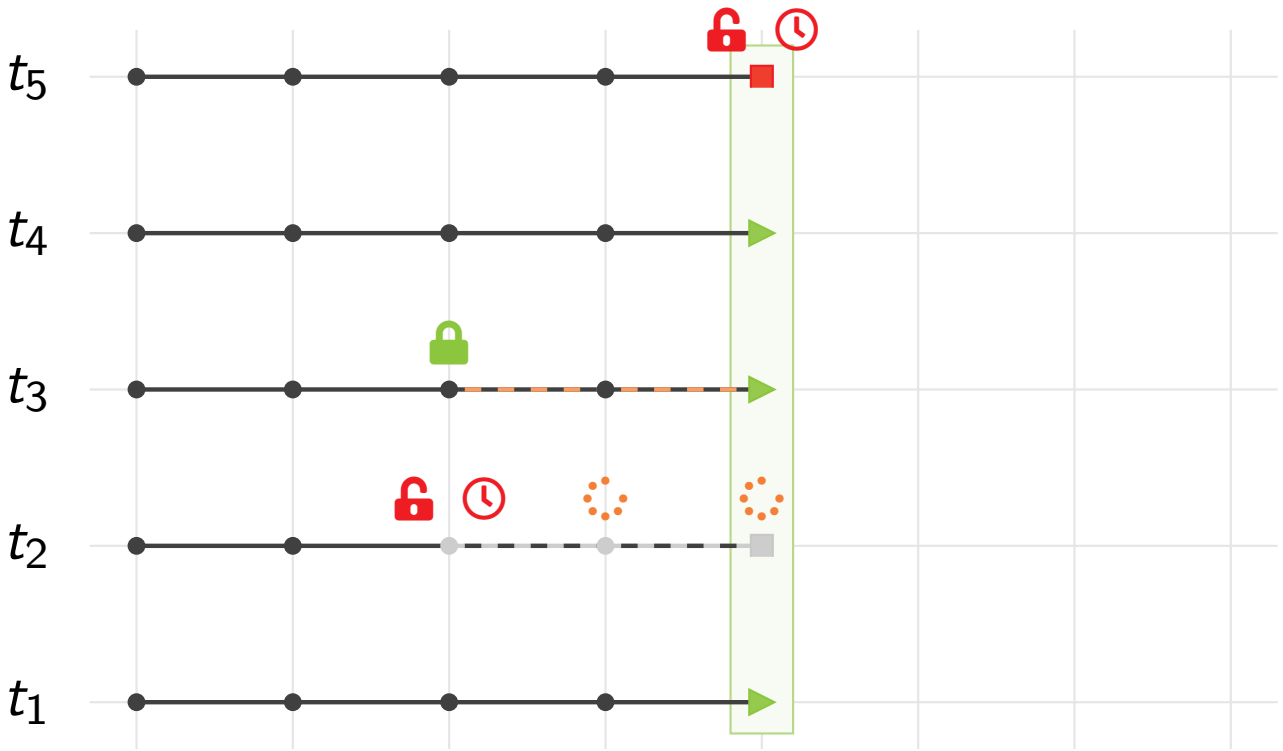
- t_5 běží
- t_4 běží
- t_3 `pthread_spin_lock()`, vstupuje do kritické sekce
- t_2 `pthread_spin_lock()`, uspí se
- t_1 běží

Spinlock



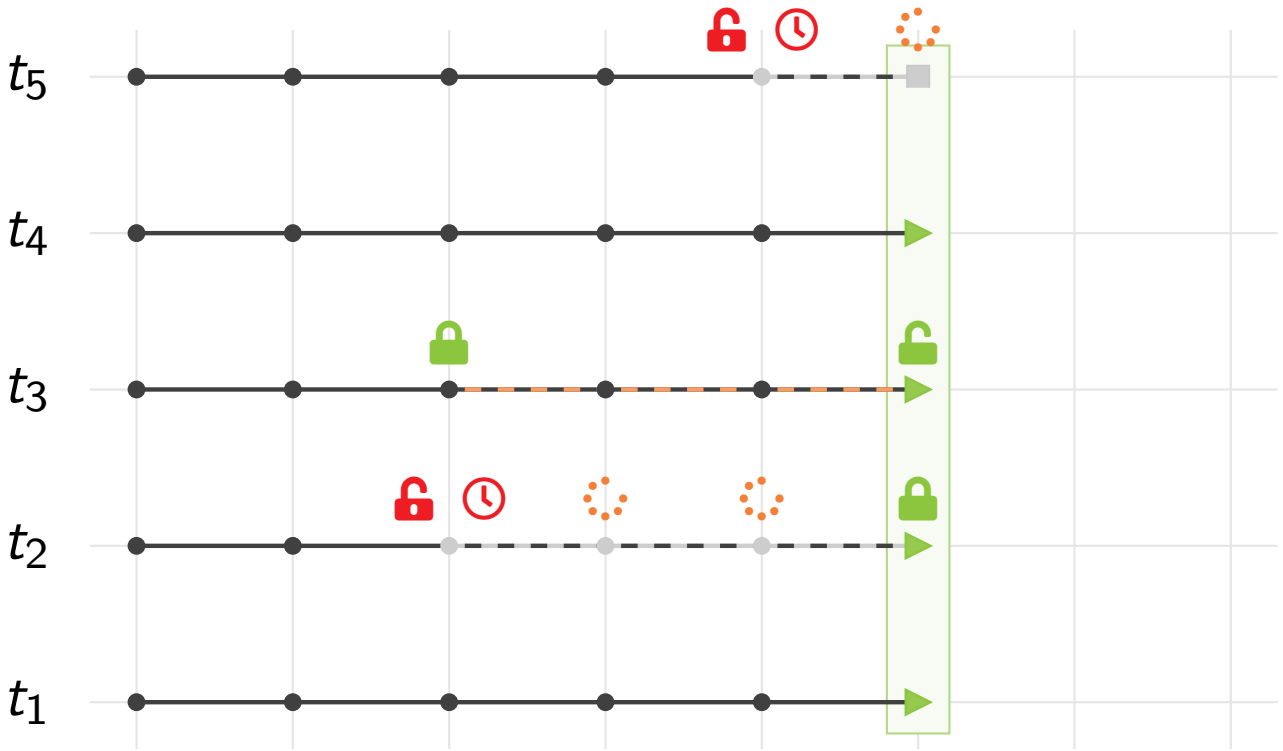
- t_5 běží
- t_4 běží
- t_3 běží v kritické sekci
- t_2 `pthread_spin_lock()`, aktivně čeká
- t_1 běží

Spinlock



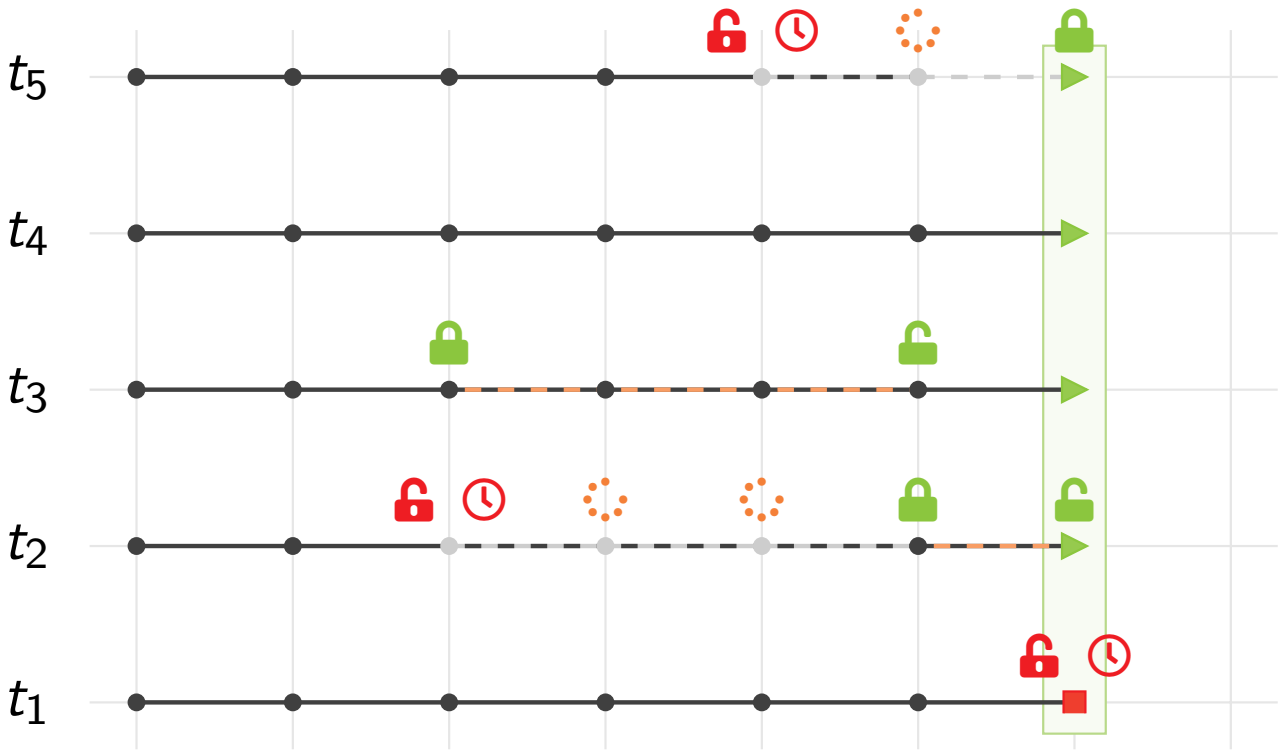
- t_5 `pthread_spin_lock()`, uspí se
- t_4 běží
- t_3 běží v kritické sekci
- t_2 `pthread_spin_lock()`, aktivně čeká
- t_1 běží

Spinlock



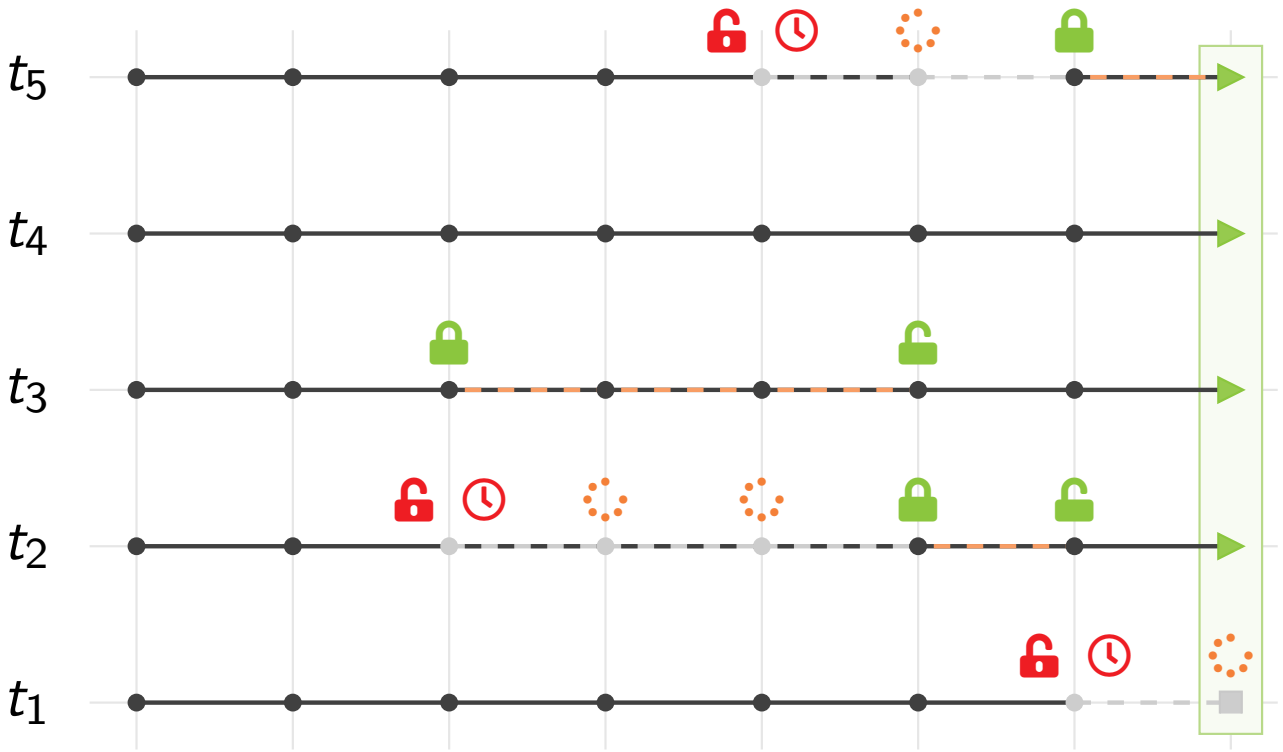
- t_5 `pthread_spin_lock()`, aktivně čeká
- t_4 běží
- t_3 `pthread_spin_unlock()`, opouští kritickou sekci
- t_2 `pthread_spin_lock()`, vstupuje do kritické sekce
- t_1 běží

Spinlock



- t_5 `pthread_spin_lock()`, vstupuje do kritické sekce
- t_4 běží
- t_3 běží
- t_2 `pthread_spin_unlock()`, opouští kritickou sekci
- t_1 `pthread_spin_lock()`, uspí se

Spinlock

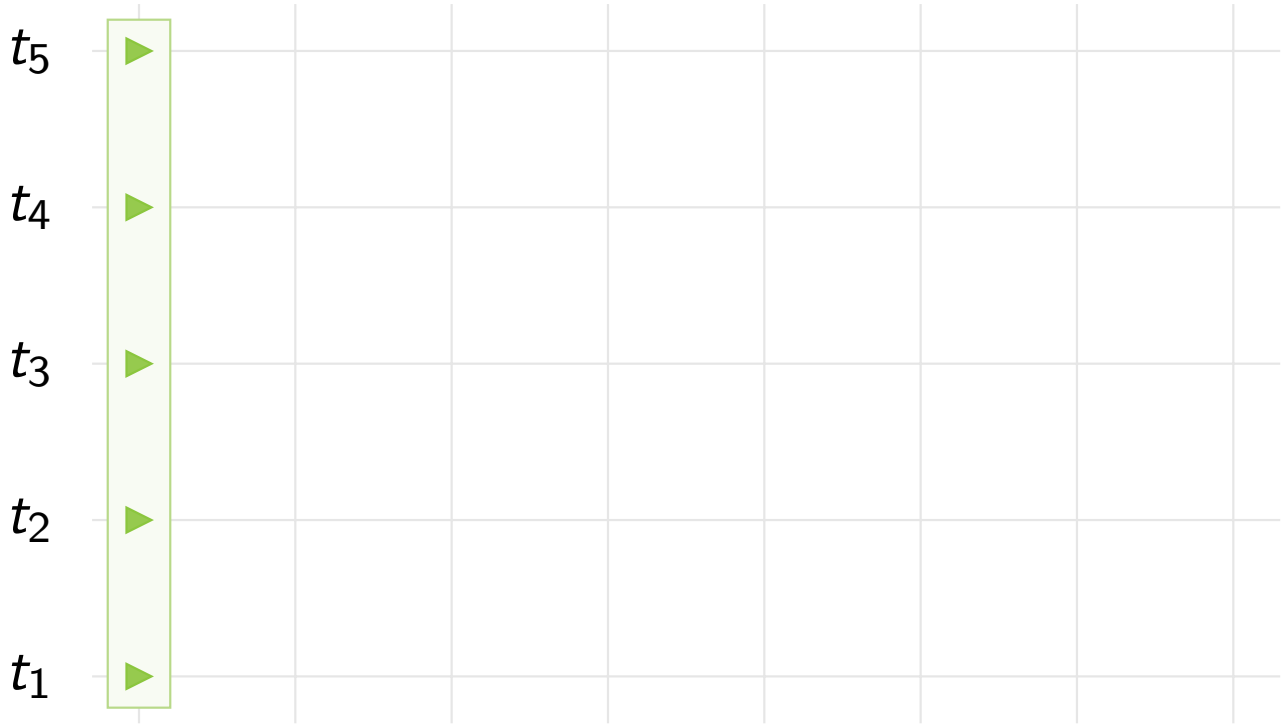


- t_5 běží v kritické sekci
- t_4 běží
- t_3 běží
- t_2 běží
- t_1 `pthread_spin_lock()`, aktivně čeká

Read-Write Zámek

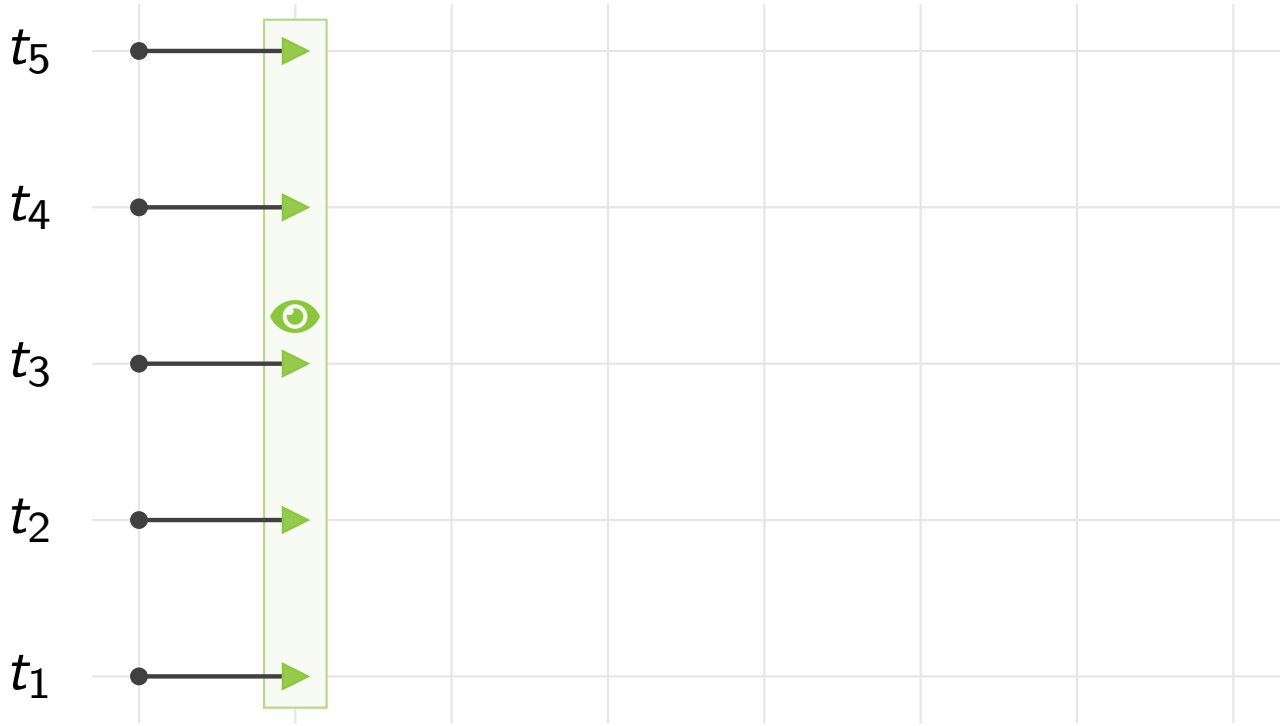
```c
void *reader(void *arg)
{
    while (!/* finished */) {
        pthread_rwlock_rdlock(&mutex);
        /* -- enter critical section -- */
        /* read data to use */
        /* -- leave critical section -- */
        pthread_rwlock_unlock(&mutex);

        /* do some computation */
    }
}

void *writer(void *arg)
{
    while (!/* finished */) {
        /* compute a new value */

        pthread_rwlock_wrlock(&mutex);
        /* -- enter critical section -- */
        /* write data for readers */
        /* -- leave critical section -- */
        pthread_rwlock_unlock(&mutex);
    }
}
```

Read-Write Zámek



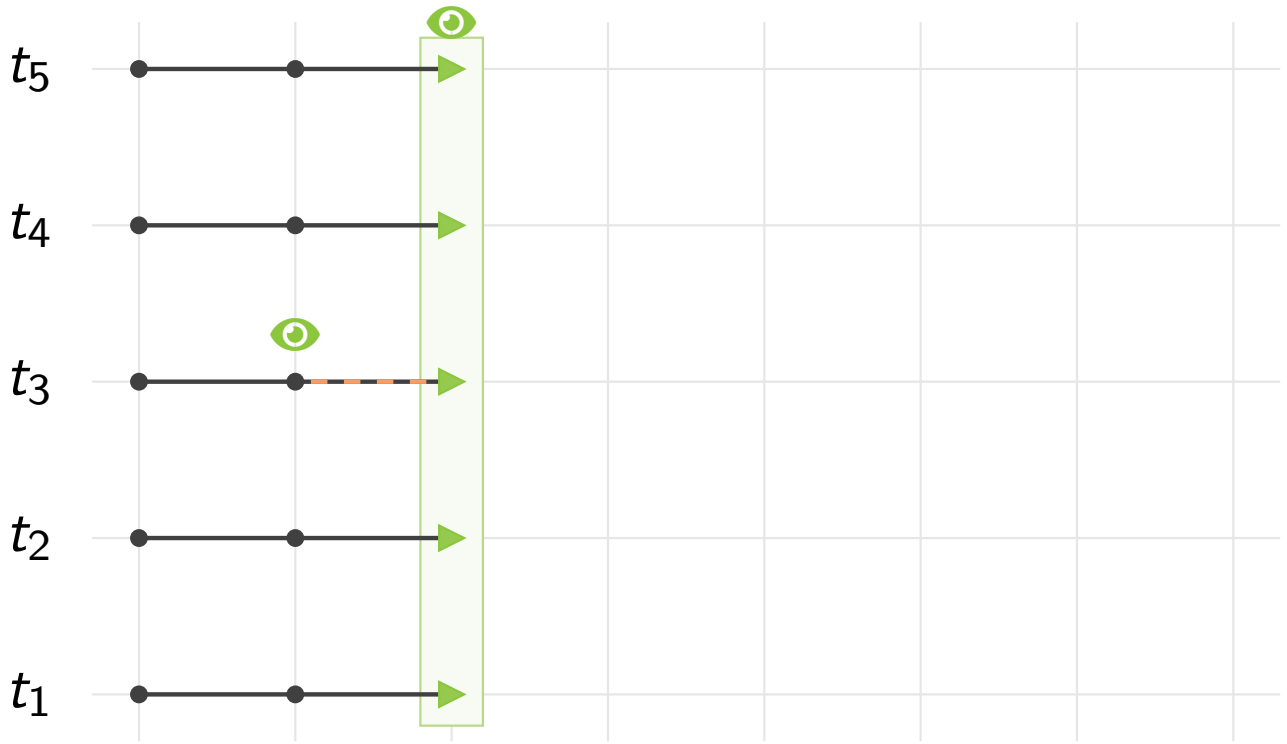
- t_5 (r) běží
- t_4 (r) běží
- t_3 (r) běží
- t_2 (w) běží
- t_1 (w) běží

Read-Write Zámek



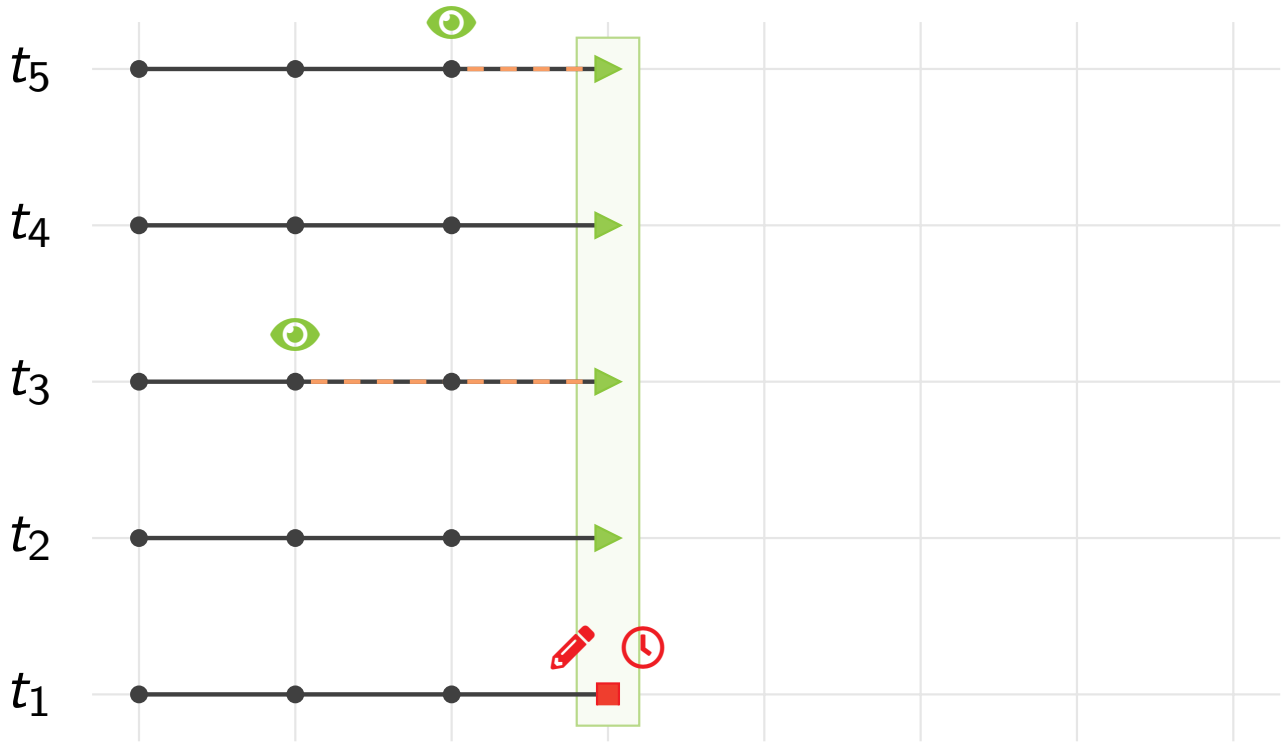
- t_5 (r) běží
- t_4 (r) běží
- t_3 (r) `pthread_rwlock_rdlock()`, vstupuje do kritické sekce
- t_2 (w) běží
- t_1 (w) běží

Read-Write Zámek



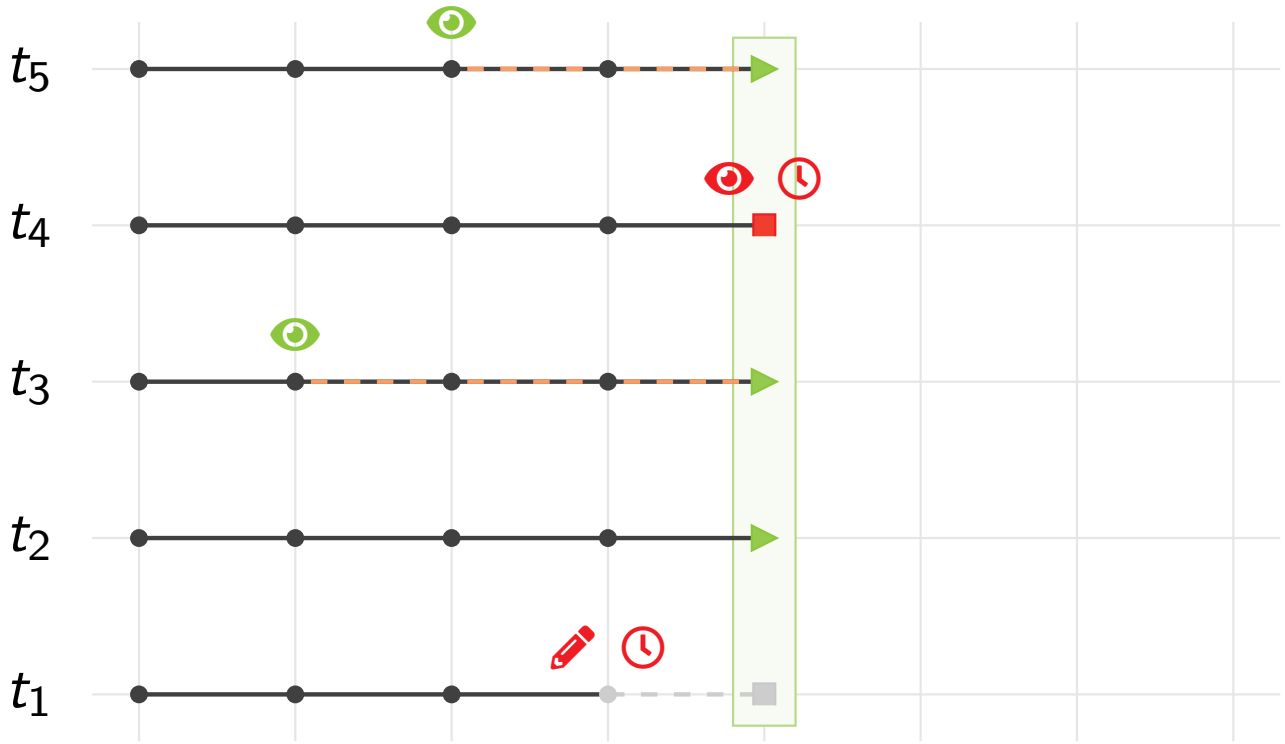
- t_5 (r) `pthread_rwlock_rdlock()`, vstupuje do kritické sekce
- t_4 (r) běží
- t_3 (r) běží v kritické sekci (pro čtění)
- t_2 (w) běží
- t_1 (w) běží

Read-Write Zámek



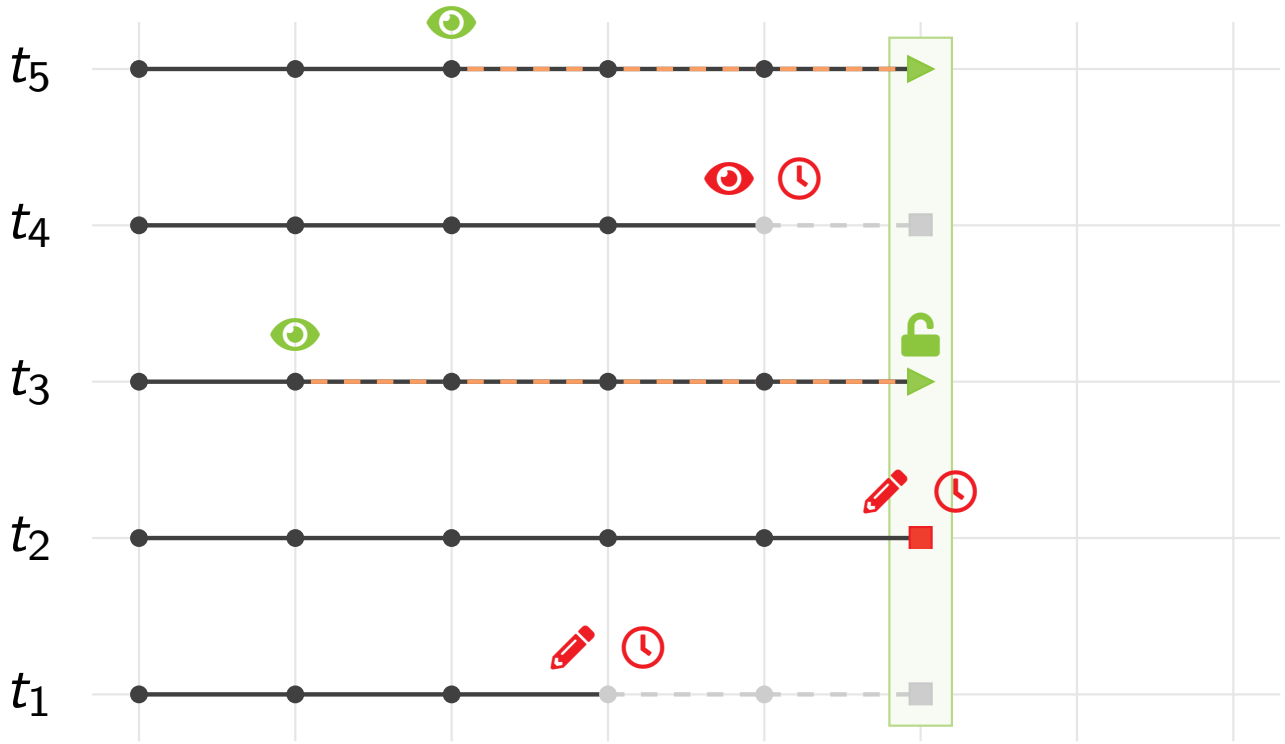
- t_5 (r) běží v kritické sekci (pro čtění)
- t_4 (r) běží
- t_3 (r) běží v kritické sekci (pro čtění)
- t_2 (w) běží
- t_1 (w) `pthread_rwlock_wrlock()`, uspí se

Read-Write Zámek



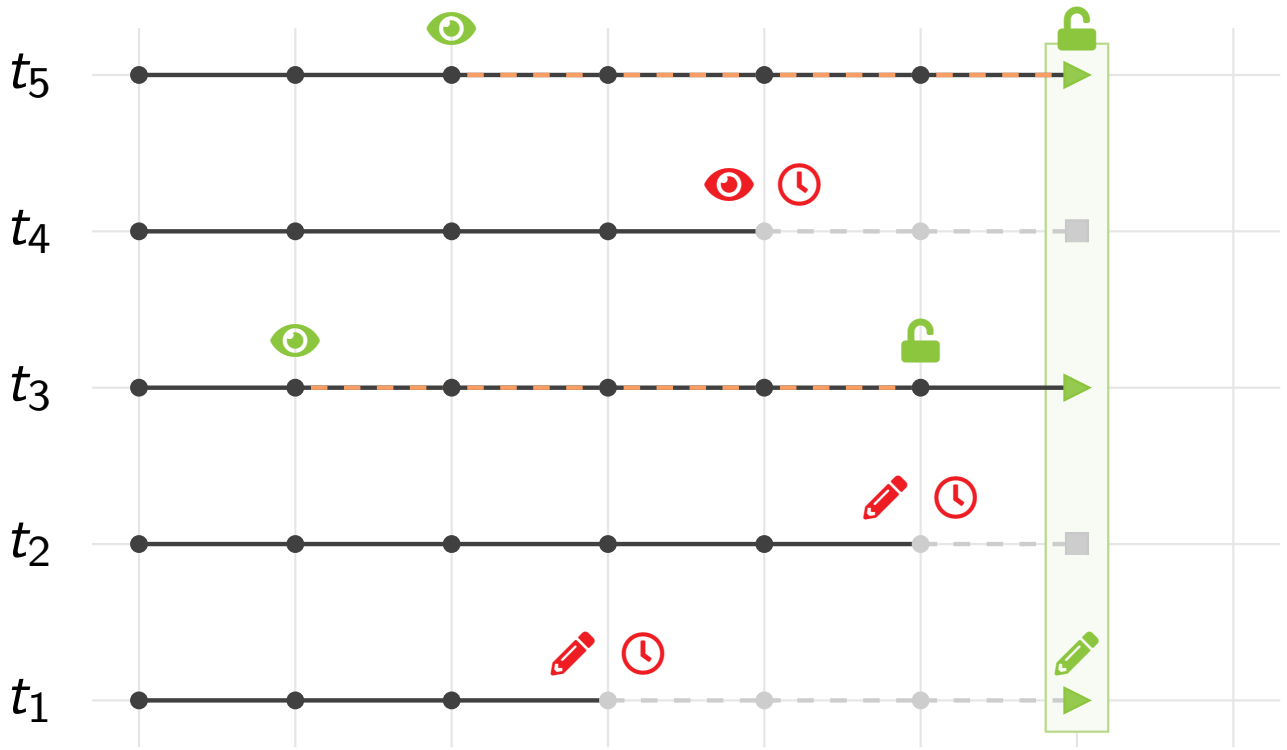
- t_5 (r) běží v kritické sekci (pro čtění)
- t_4 (r) `pthread_rwlock_rdlock()`, uspí se
- t_3 (r) běží v kritické sekci (pro čtění)
- t_2 (w) běží
- t_1 (w) `pthread_rwlock_wrlock()`, čeká

Read-Write Zámek



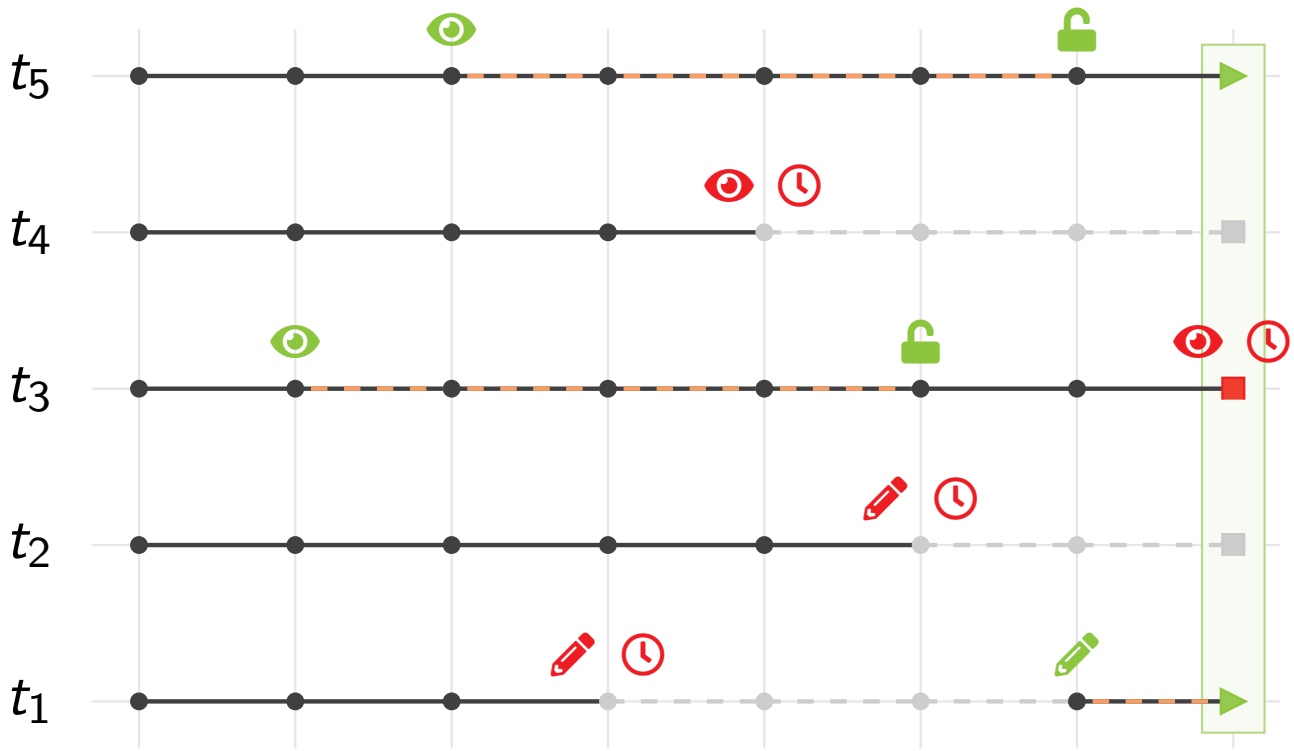
- t_5 (r) běží v kritické sekci (pro čtění)
- t_4 (r) `pthread_rwlock_rdlock()`, čeká
- t_3 (r) `pthread_rwlock_unlock()`, opouští kritickou sekci (pro čtění)
- t_2 (w) `pthread_rwlock_wrlock()`, uspí se
- t_1 (w) `pthread_rwlock_wrlock()`, čeká

Read-Write Zámek



- t_5 (r) `pthread_rwlock_unlock()`, opouští kritickou sekci (pro čtění)
- t_4 (r) `pthread_rwlock_rdlock()`, čeká
- t_3 (r) běží
- t_2 (w) `pthread_rwlock_wrlock()`, čeká
- t_1 (w) `pthread_rwlock_wrlock()`, vstupuje do kritické sekce

Read-Write Zámek

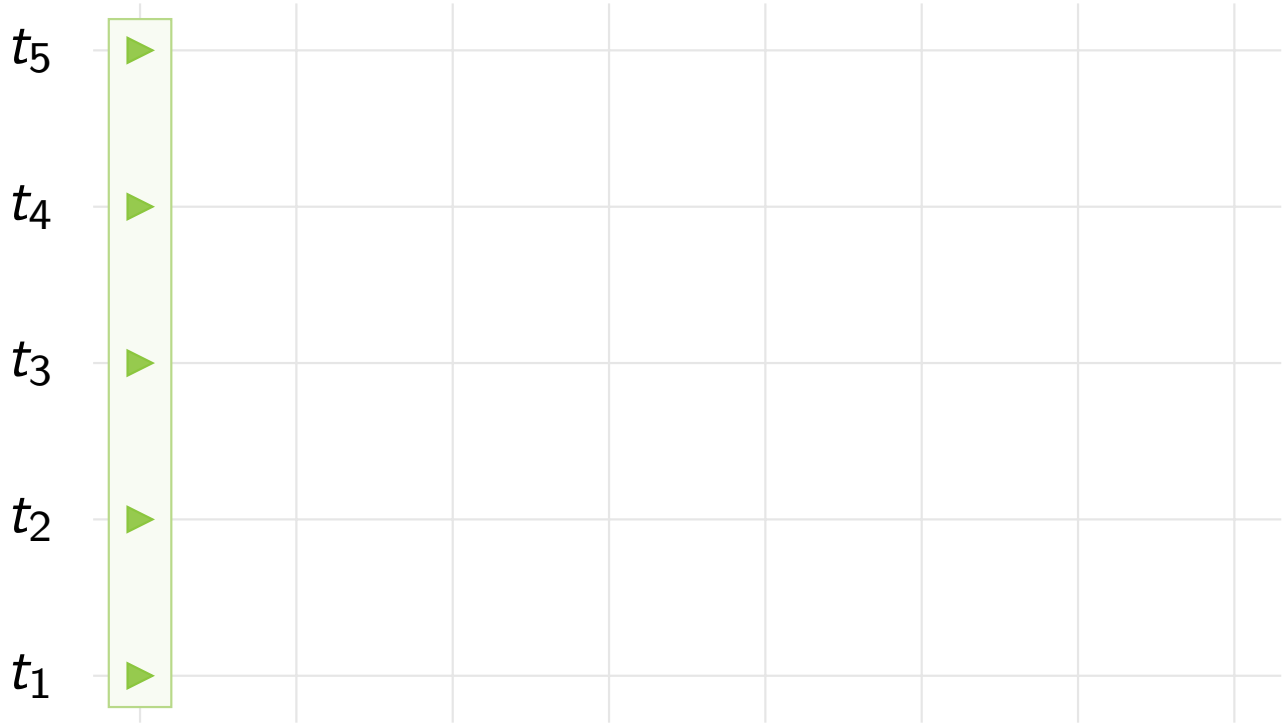


- t_5 (r) běží
- t_4 (r) `pthread_rwlock_rdlock()`, čeká
- t_3 (r) `pthread_rwlock_rdlock()`, uspí se
- t_2 (w) `pthread_rwlock_wrlock()`, čeká
- t_1 (w) běží v kritické sekci (pro zápis)

Bariéra

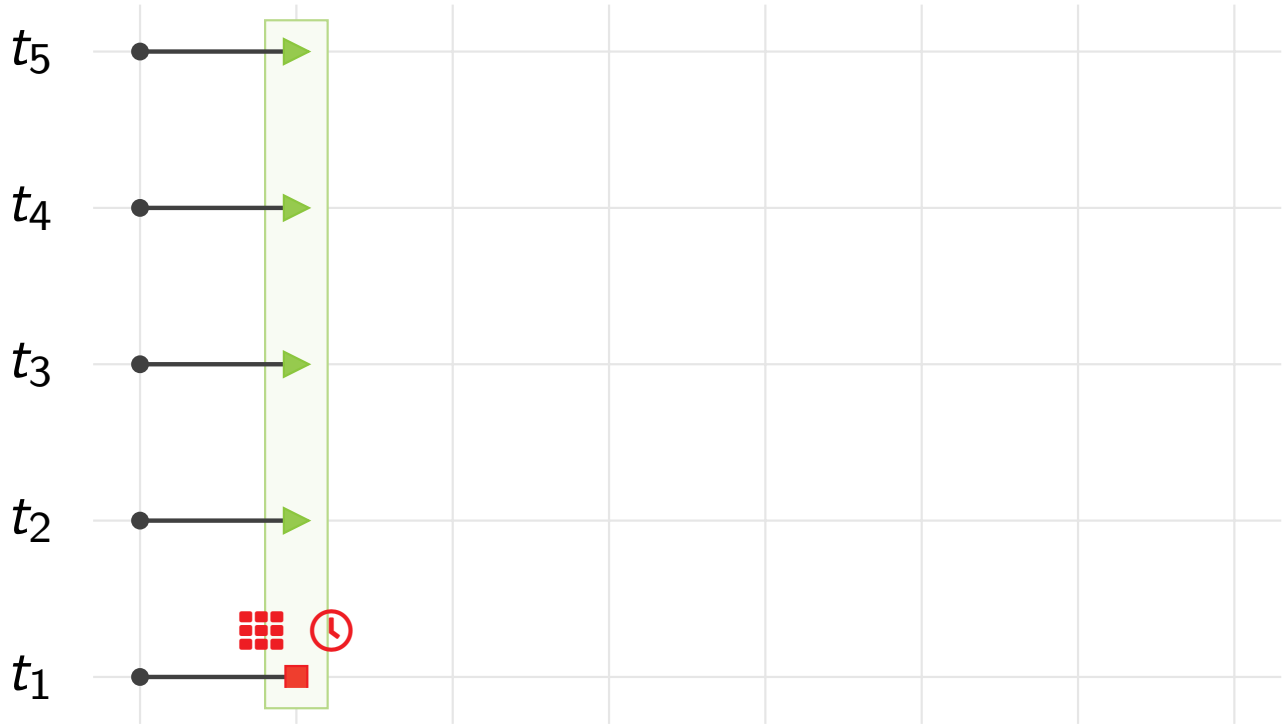
```c
volatile struct worker_state *state;

void *worker(void *arg)
{
    /* initialize global state for this thread */

    /* wait for other threads to finish intialization */
    pthread_barrier_wait(&barrier);

    /* everything is initialized now */

    while (!/* finished */) {
        /* do some work */
    }
}

int main(void) {
    pthread_barrier_init(&barrier, N + 1); // N threads + main

    /* allocate memory for state */

    for (int i = 0; i < N; ++N) {
        pthread_create(&thread[i], NULL, worker, NULL);
    }

    /* wait for threads to initialize their state */
    pthread_barrier(&barrier);

    /* inspect global state etc. */
    /* ... */
}
```

Bariéra



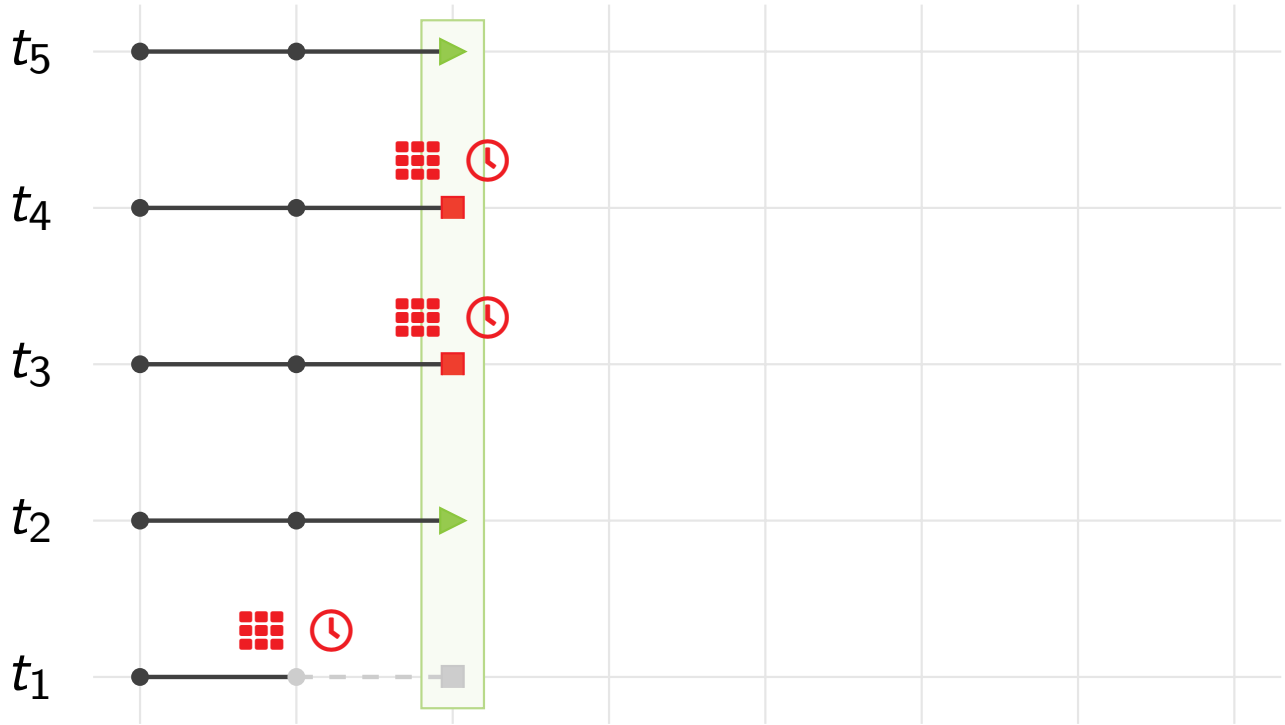
- t_5 běží
- t_4 běží
- t_3 běží
- t_2 běží
- t_1 běží

Bariéra



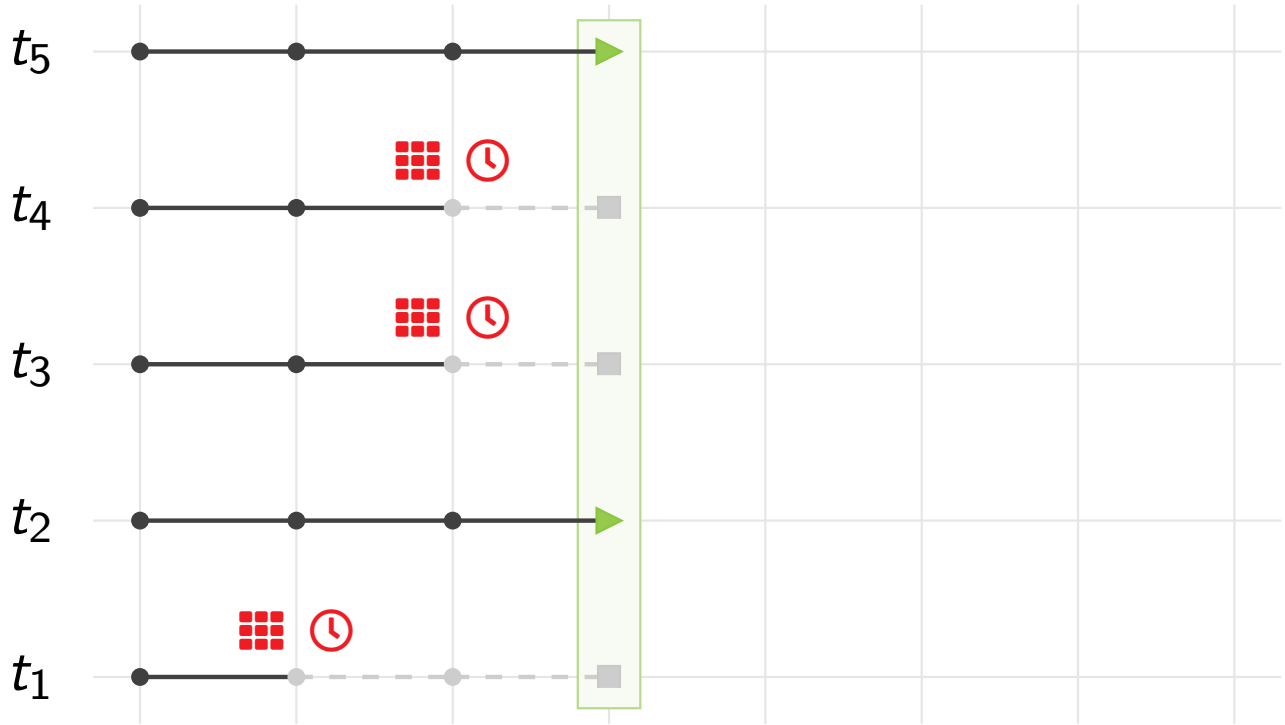
- t_5 běží
- t_4 běží
- t_3 běží
- t_2 běží
- t_1 `pthread_barrier_wait()` (1 z 5 vláken), uspí se

Bariéra



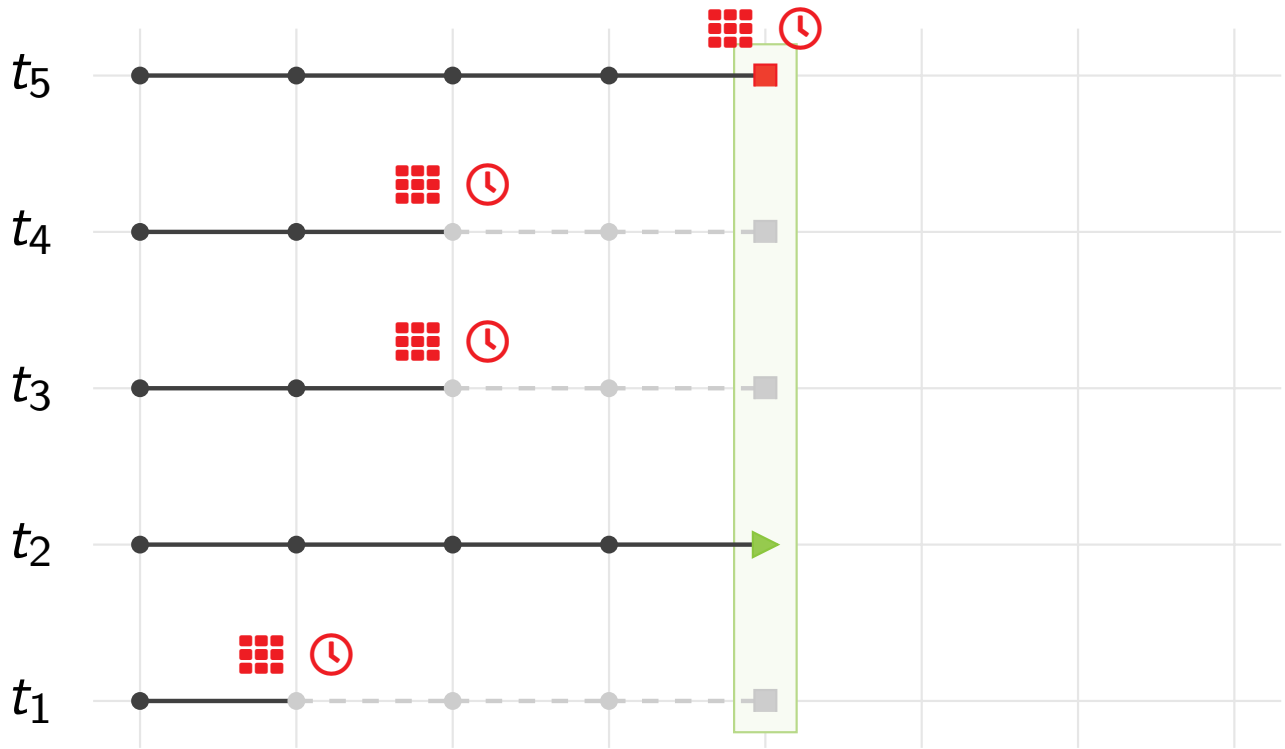
- t_5 běží
- t_4 `pthread_barrier_wait()` (3 z 5 vláken), uspí se
- t_3 `pthread_barrier_wait()` (3 z 5 vláken), uspí se
- t_2 běží
- t_1 `pthread_barrier_wait()` (3 z 5 vláken), čeká

Bariéra



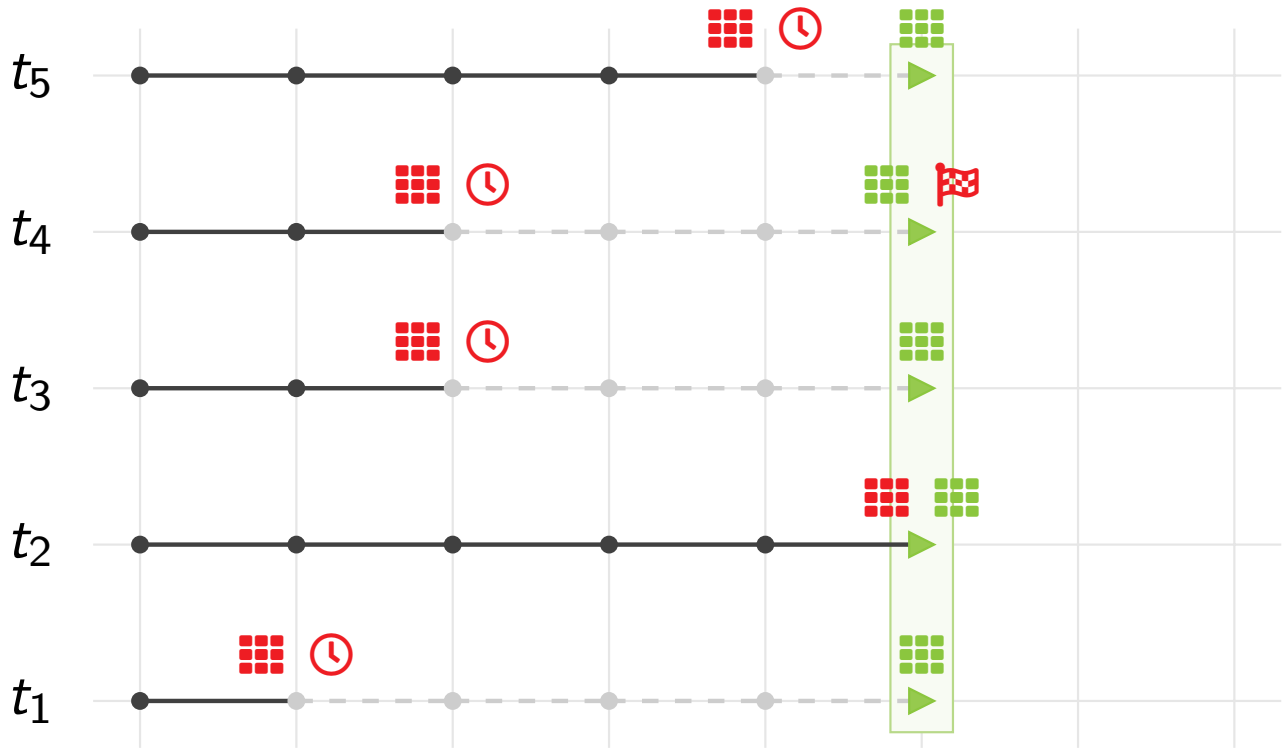
- t_5 běží
- t_4 `pthread_barrier_wait()` (3 z 5 vláken), čeká
- t_3 `pthread_barrier_wait()` (3 z 5 vláken), čeká
- t_2 běží
- t_1 `pthread_barrier_wait()` (3 z 5 vláken), čeká

Bariéra



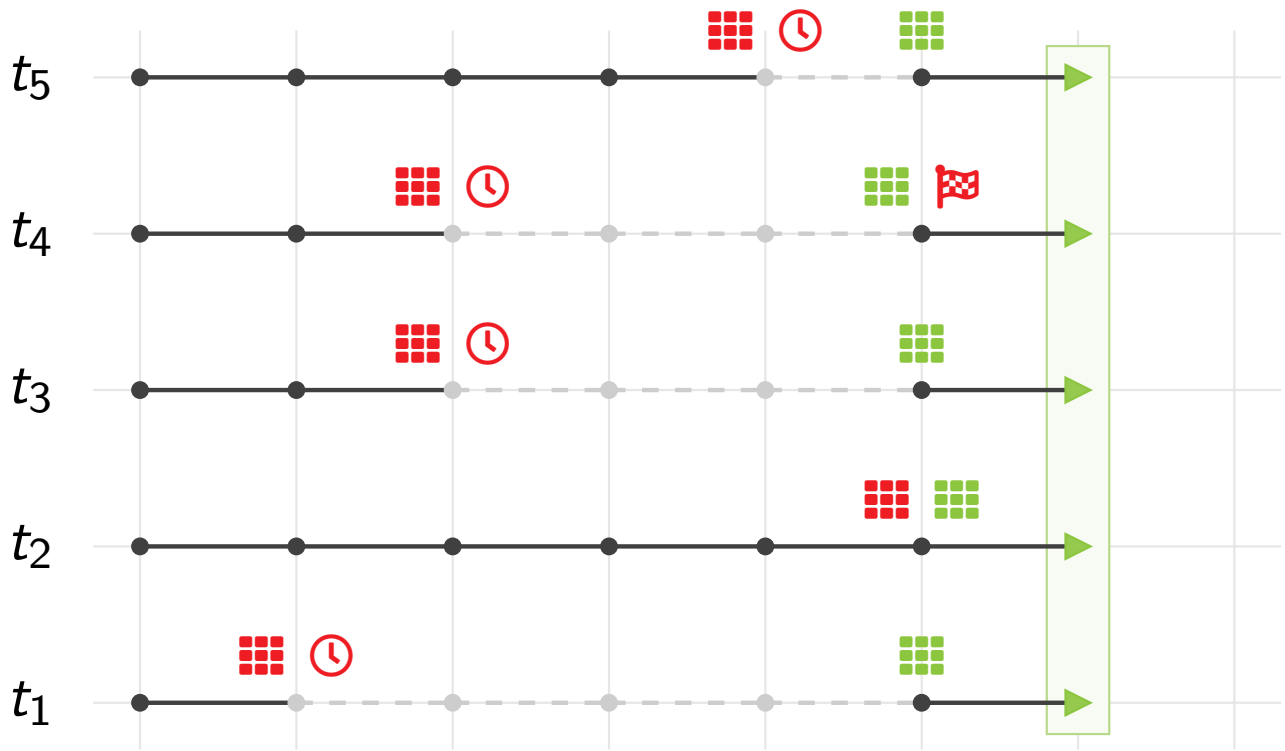
- t_5 `pthread_barrier_wait()` (4 z 5 vláken), uspí se
- t_4 `pthread_barrier_wait()` (4 z 5 vláken), čeká
- t_3 `pthread_barrier_wait()` (4 z 5 vláken), čeká
- t_2 běží
- t_1 `pthread_barrier_wait()` (4 z 5 vláken), čeká

Bariéra



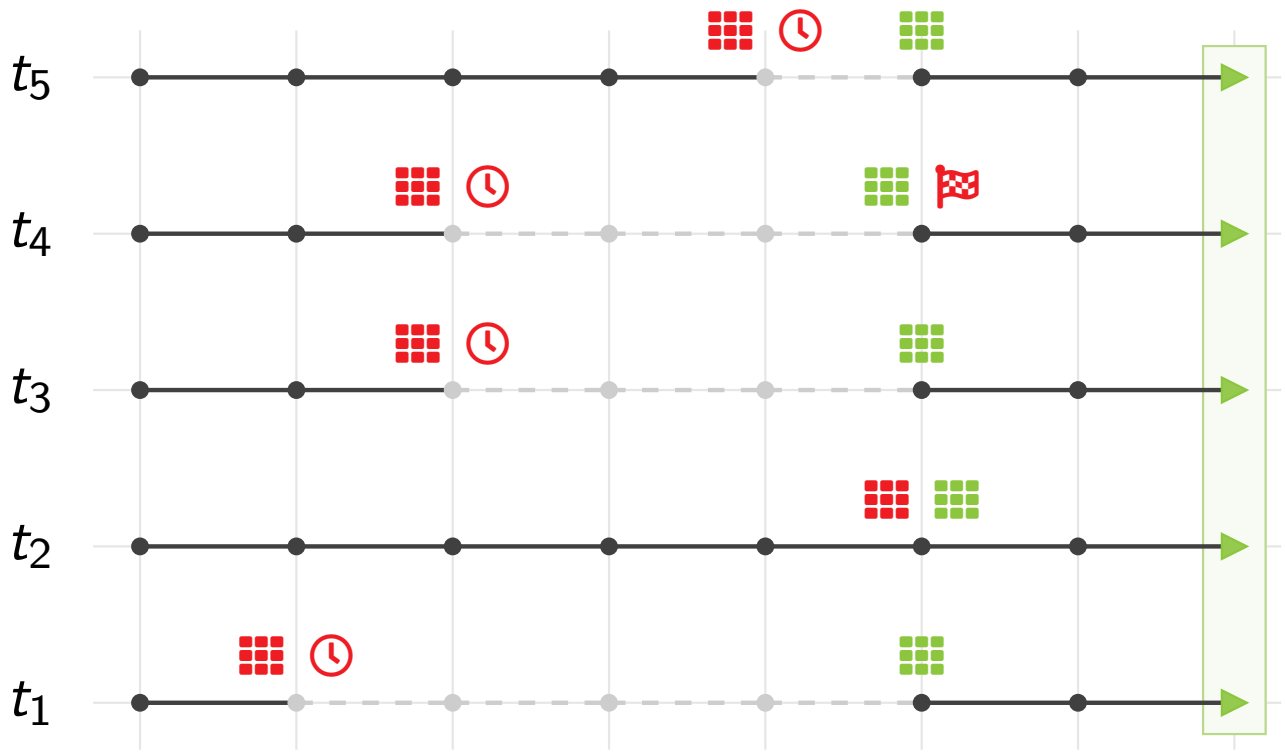
- t_5 `pthread_barrier_wait() = 0`, probudí se
- t_4 `pthread_barrier_wait() = PTHREAD_BARRIER_SERIAL_THREAD`, probudí se
- t_3 `pthread_barrier_wait() = 0`, probudí se
- t_2 `pthread_barrier_wait() = 0`, pokračuje
- t_1 `pthread_barrier_wait() = 0`, probudí se

Bariéra



- t_5 běží
- t_4 běží
- t_3 běží
- t_2 běží
- t_1 běží

Bariéra



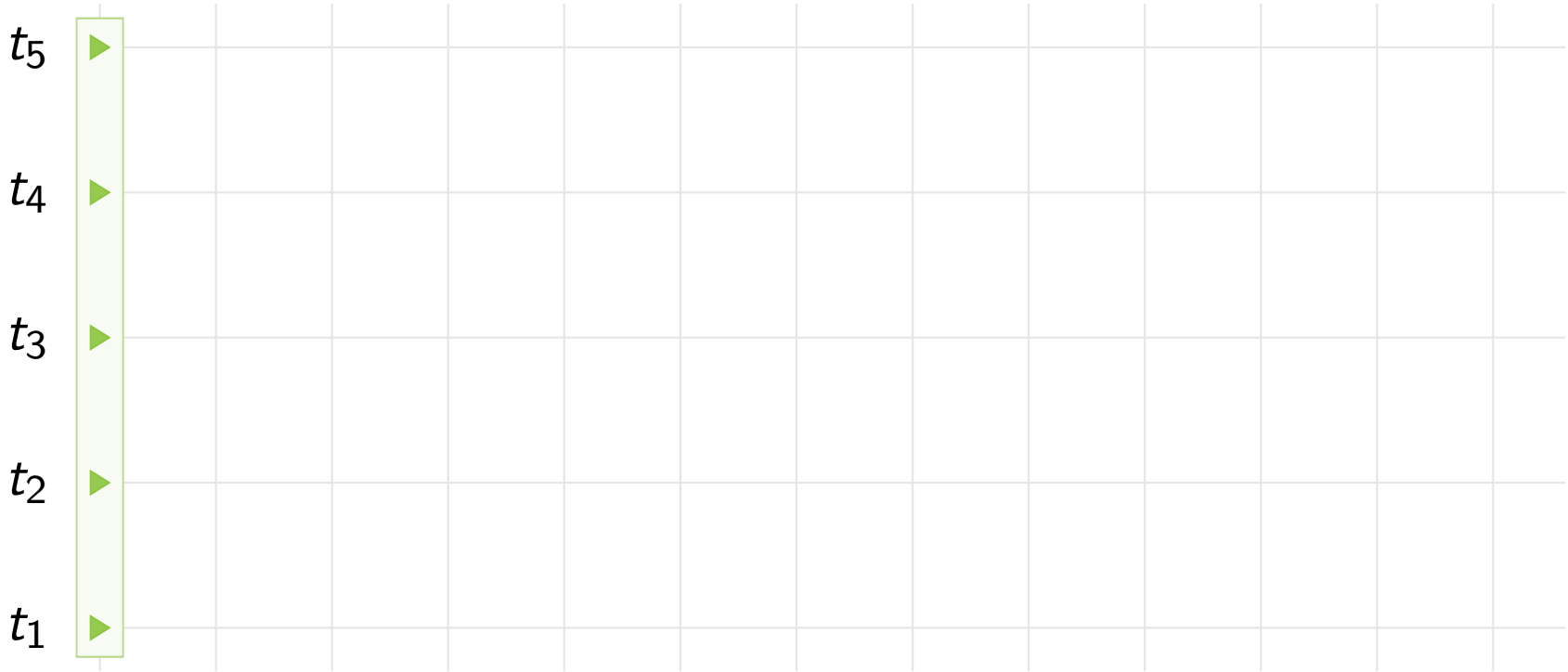
- t_5 běží
- t_4 běží
- t_3 běží
- t_2 běží
- t_1 běží

Podmínková proměnná

```c
void *producer(void *arg)
{
    while (!/* finished */) {
        /* produce a task */

        pthread_mutex_lock(&mutex);
        queue_push(queue, task);

        /* we don't ALWAYS need to call signal here -- why? */
        pthread_cond_signal(&cond);
        pthread_mutex_unlock(&mutex);
    }
}

void *consumer(void *arg) {
    while (!/* finished */) {
        mutex_lock(&mutex);
        while (queue_empty(queue)) {
            pthread_cond_wait(&cond, &mutex);
        }

        task = queue_unshift(queue);
        pthread_mutex_unlock(&mutex);

        /* process the task */
    }
}
```

Podmínková proměnná

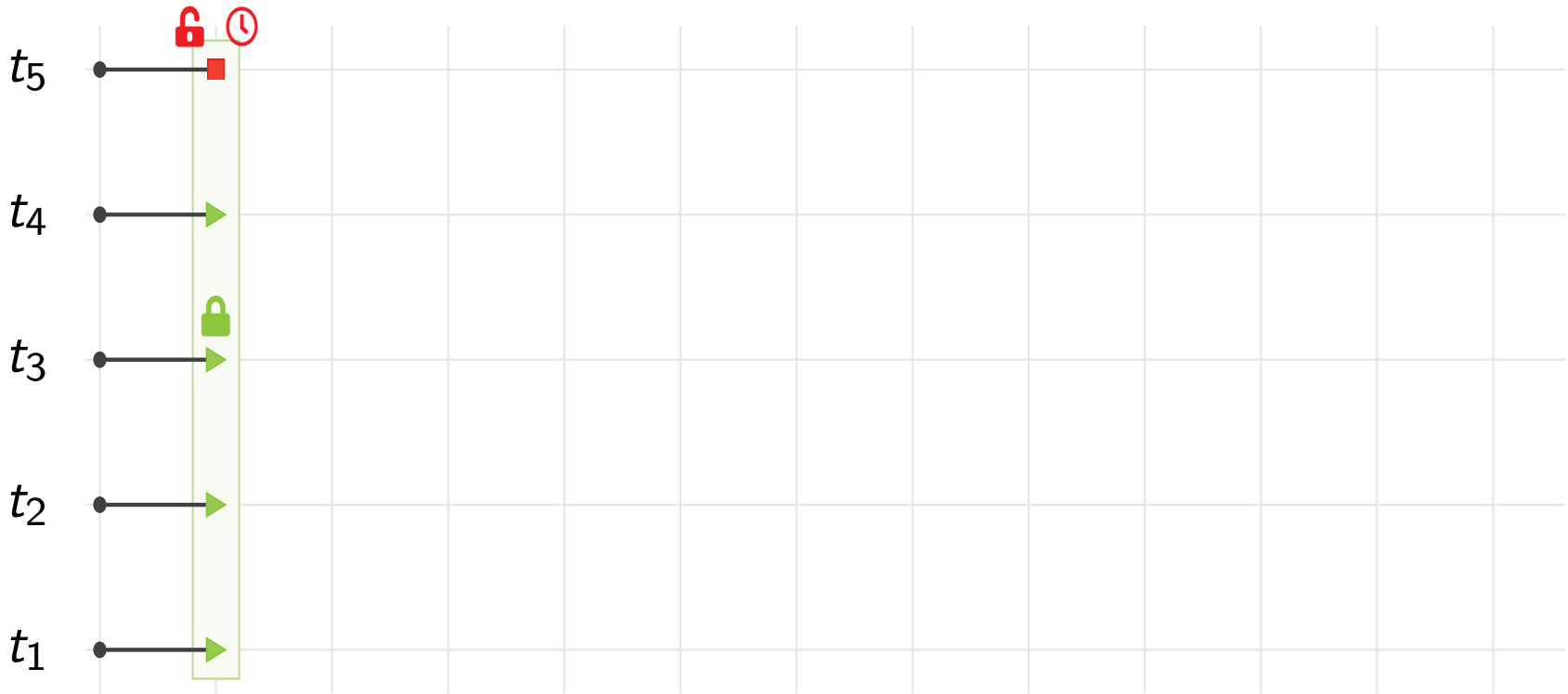
Ukázka 1



- t_5 (c) běží
- t_4 (c) běží
- t_3 (c) běží
- t_2 (p) běží
- t_1 (p) běží

Podmínková proměnná

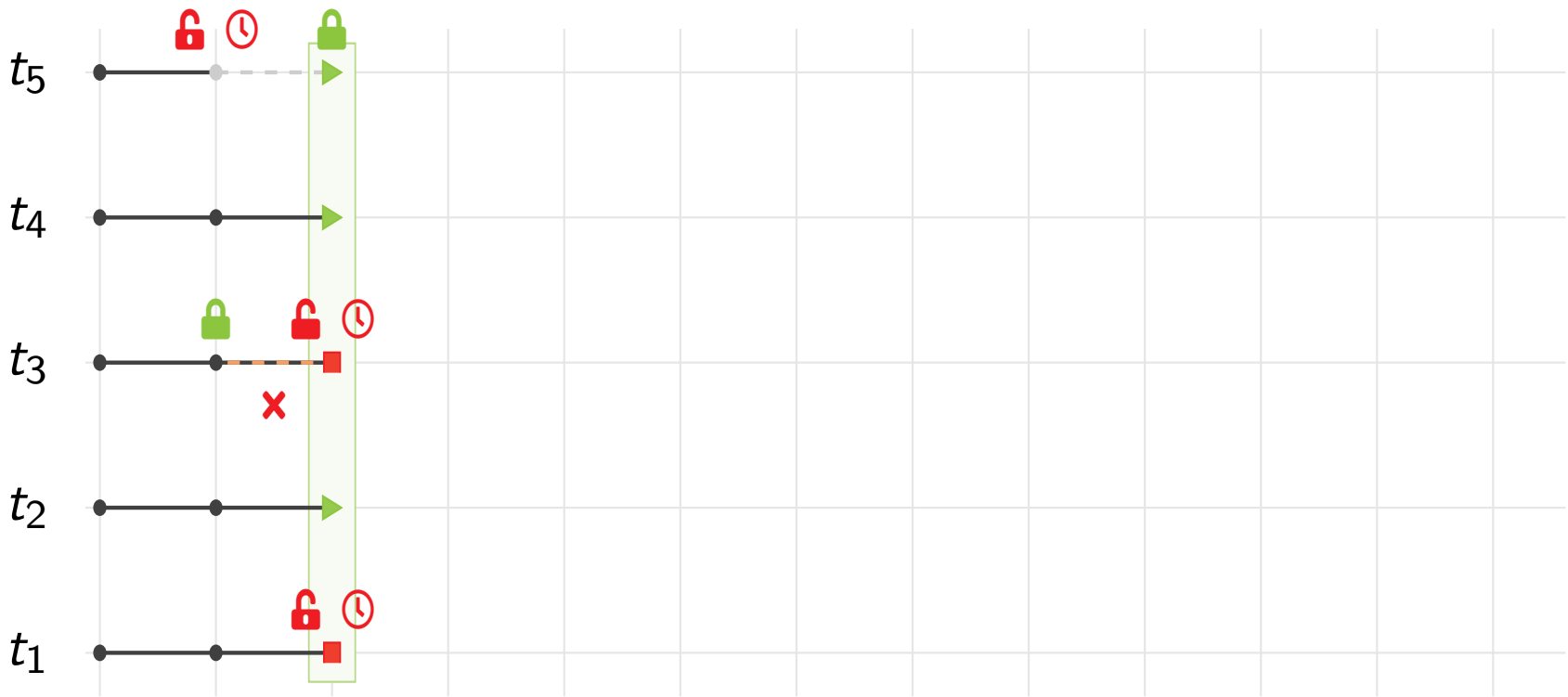
Ukázka 1



- t_5 (c) `pthread_mutex_lock()`, uspí se
- t_4 (c) běží
- t_3 (c) `pthread_mutex_lock()`, vstupuje do kritické sekce
- t_2 (p) běží
- t_1 (p) běží

Podmínková proměnná

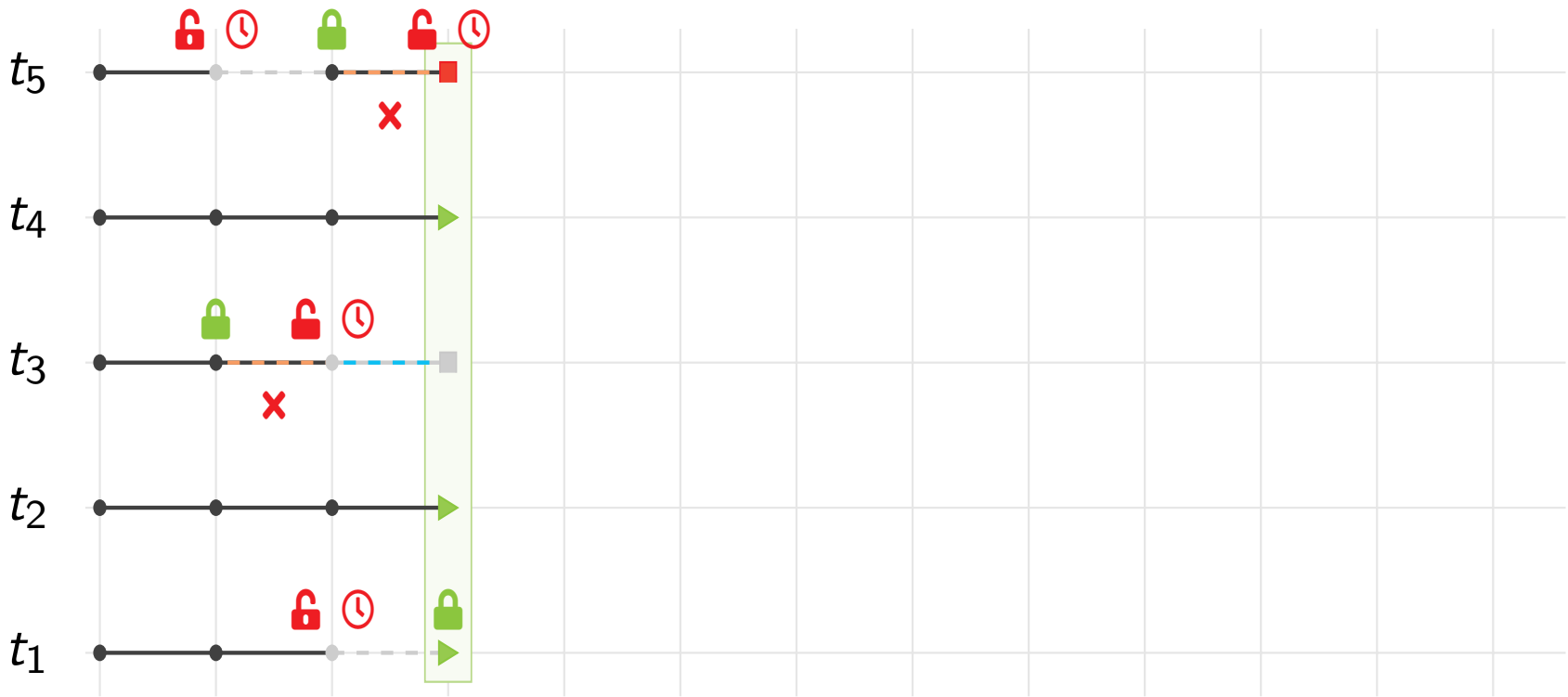
Ukázka 1



- t_5 (c) `pthread_mutex_lock()`, probudí se
- t_4 (c) běží
- t_3 (c) podmínka neplatí $\rightarrow$ `pthread_cond_wait()`, uvolní zámek
- t_2 (p) běží
- t_1 (p) `pthread_mutex_lock()`, uspí se

Podmínková proměnná

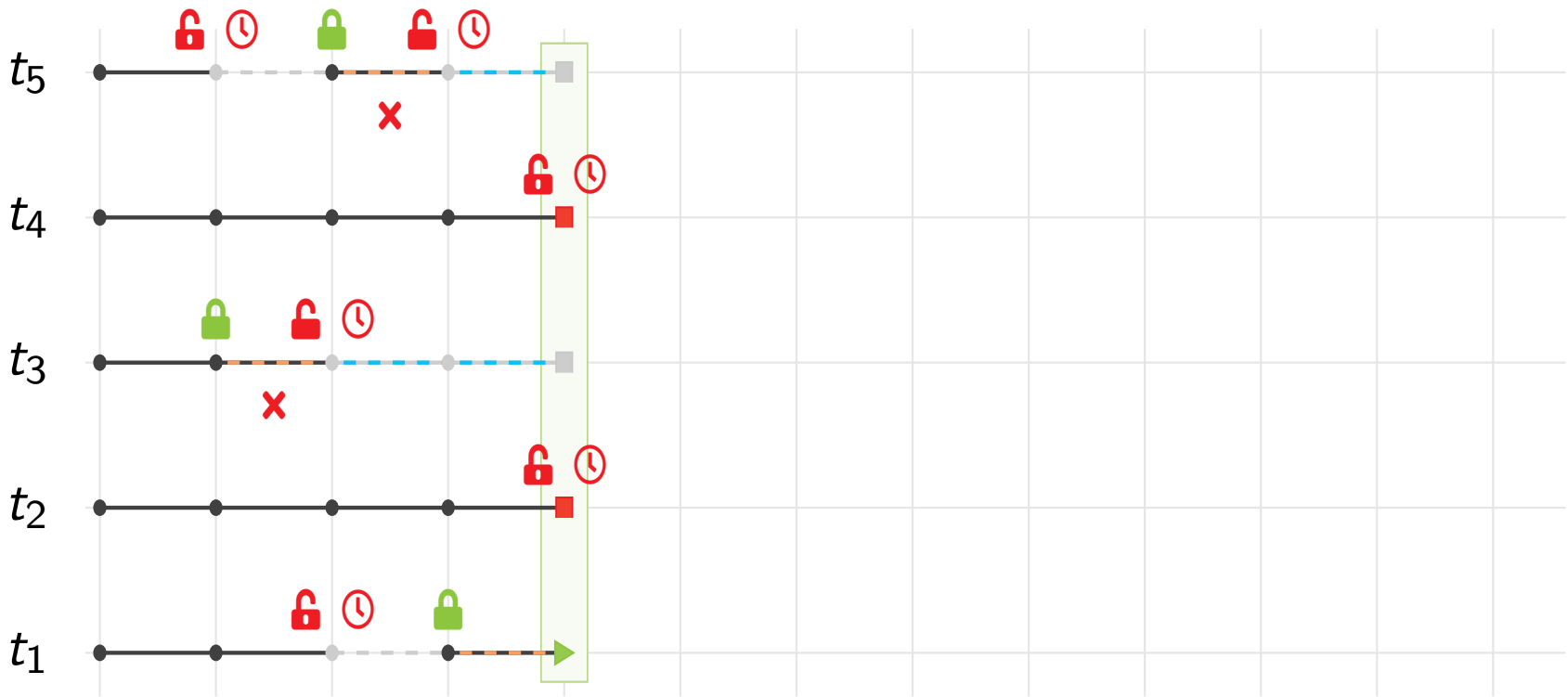
Ukázka 1



- t_5 (c) podmínka neplatí → `pthread_cond_wait()`, uvolní zámek
- t_4 (c) běží
- t_3 (c) `pthread_cond_wait()`, čeká
- t_2 (p) běží
- t_1 (p) `pthread_mutex_lock()`, vstupuje do kritické sekce

Podmínková proměnná

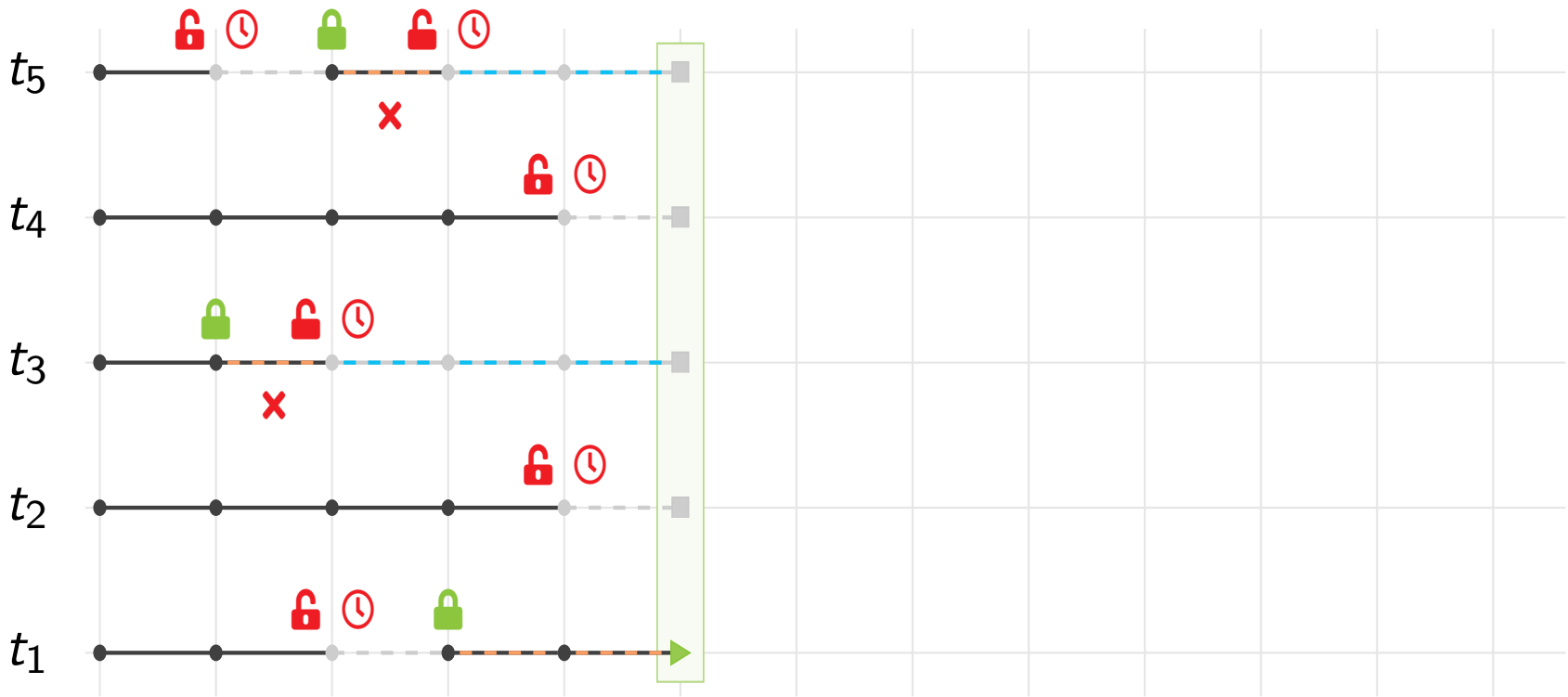
Ukázka 1



- t_5 (c) `pthread_cond_wait()`, čeká
- t_4 (c) `pthread_mutex_lock()`, uspí se
- t_3 (c) `pthread_cond_wait()`, čeká
- t_2 (p) `pthread_mutex_lock()`, uspí se
- t_1 (p) běží v kritické sekci (vkládá prvek)

Podmínková proměnná

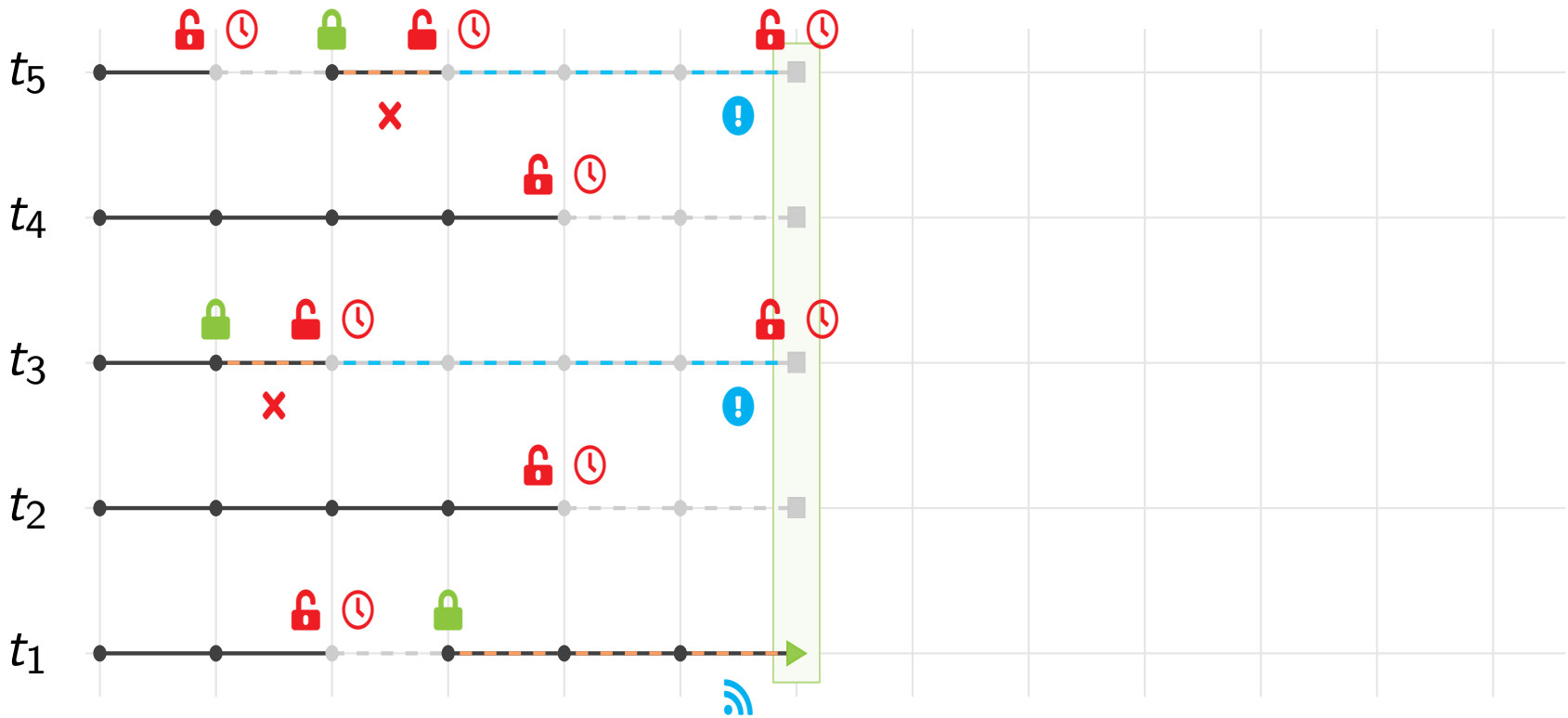
Ukázka 1



- t_5 (c) `pthread_cond_wait()`, čeká
- t_4 (c) `pthread_mutex_lock()`, čeká
- t_3 (c) `pthread_cond_wait()`, čeká
- t_2 (p) `pthread_mutex_lock()`, čeká
- t_1 (p) běží v kritické sekci (vkládá prvek)

Podmínková proměnná

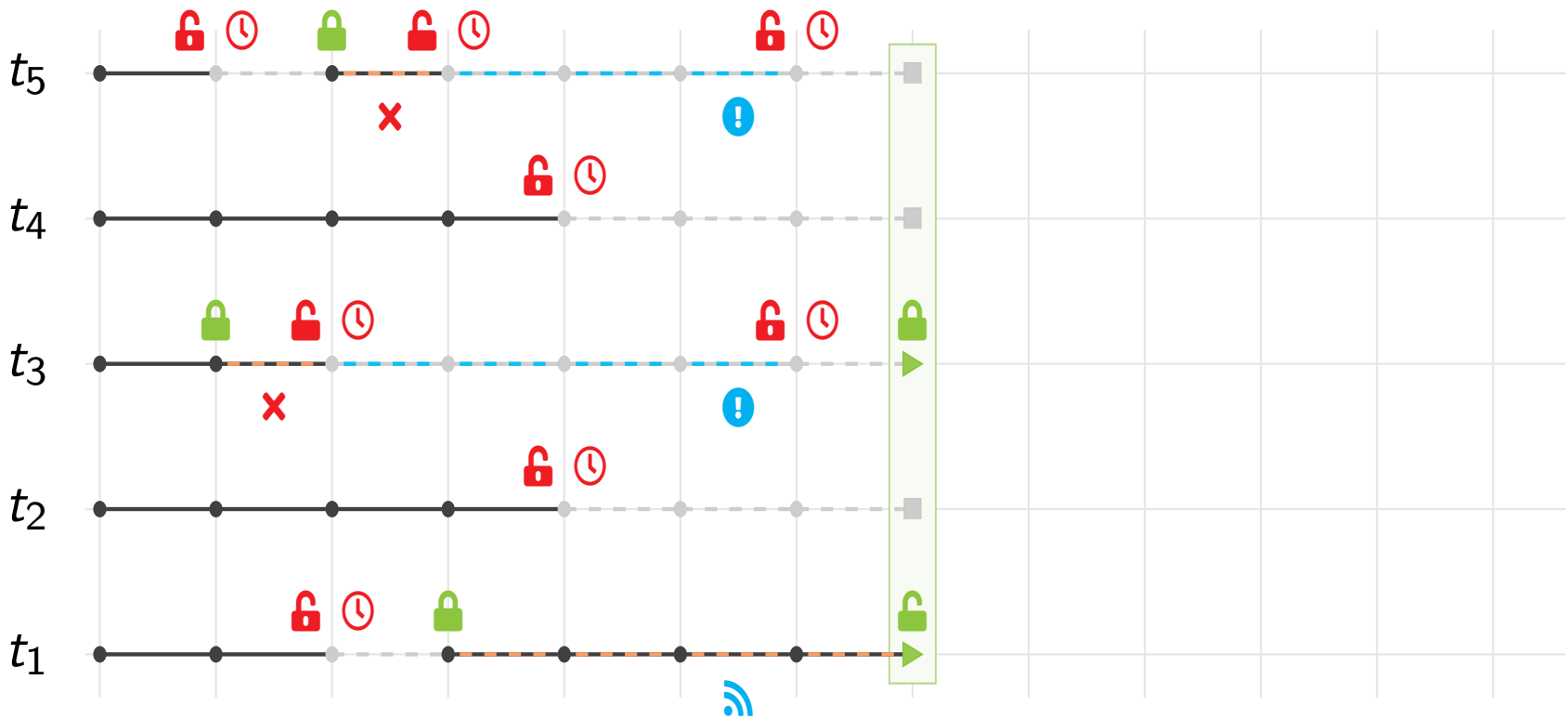
Ukázka 1



- t_5 (c) `pthread_cond_wait()` → `pthread_mutex_lock()`, uspí se
- t_4 (c) `pthread_mutex_lock()`, čeká
- t_3 (c) `pthread_cond_wait()` → `pthread_mutex_lock()`, uspí se
- t_2 (p) `pthread_mutex_lock()`, čeká
- t_1 (p) `pthread_cond_signal()`, běží v kritické sekci

Podmínková proměnná

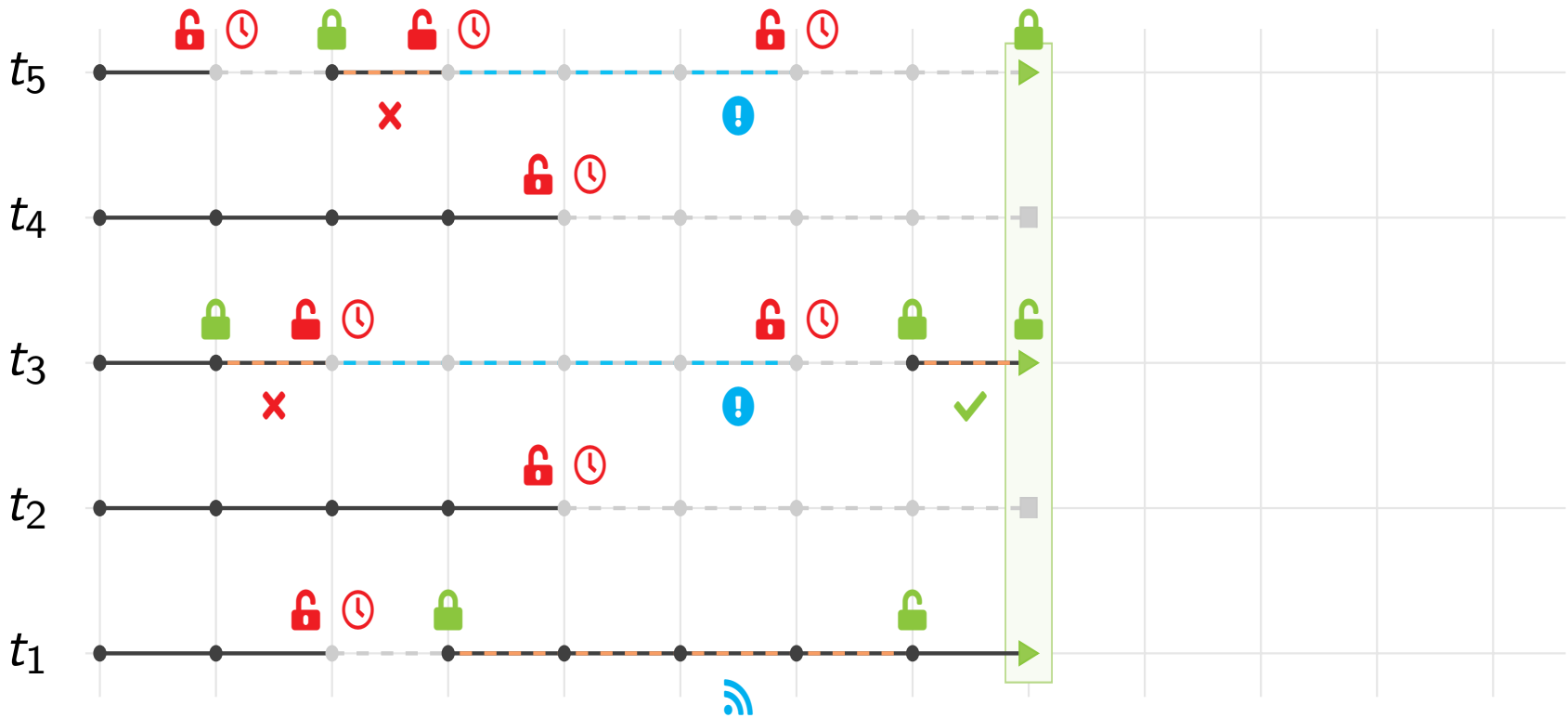
Ukázka 1



- t_5 (c) `pthread_mutex_lock()`, čeká
- t_4 (c) `pthread_mutex_lock()`, čeká
- t_3 (c) `pthread_mutex_lock()`, vstupuje do kritické sekce
- t_2 (p) `pthread_mutex_lock()`, čeká
- t_1 (p) `pthread_mutex_unlock()`, opouští kritickou sekci

Podmínková proměnná

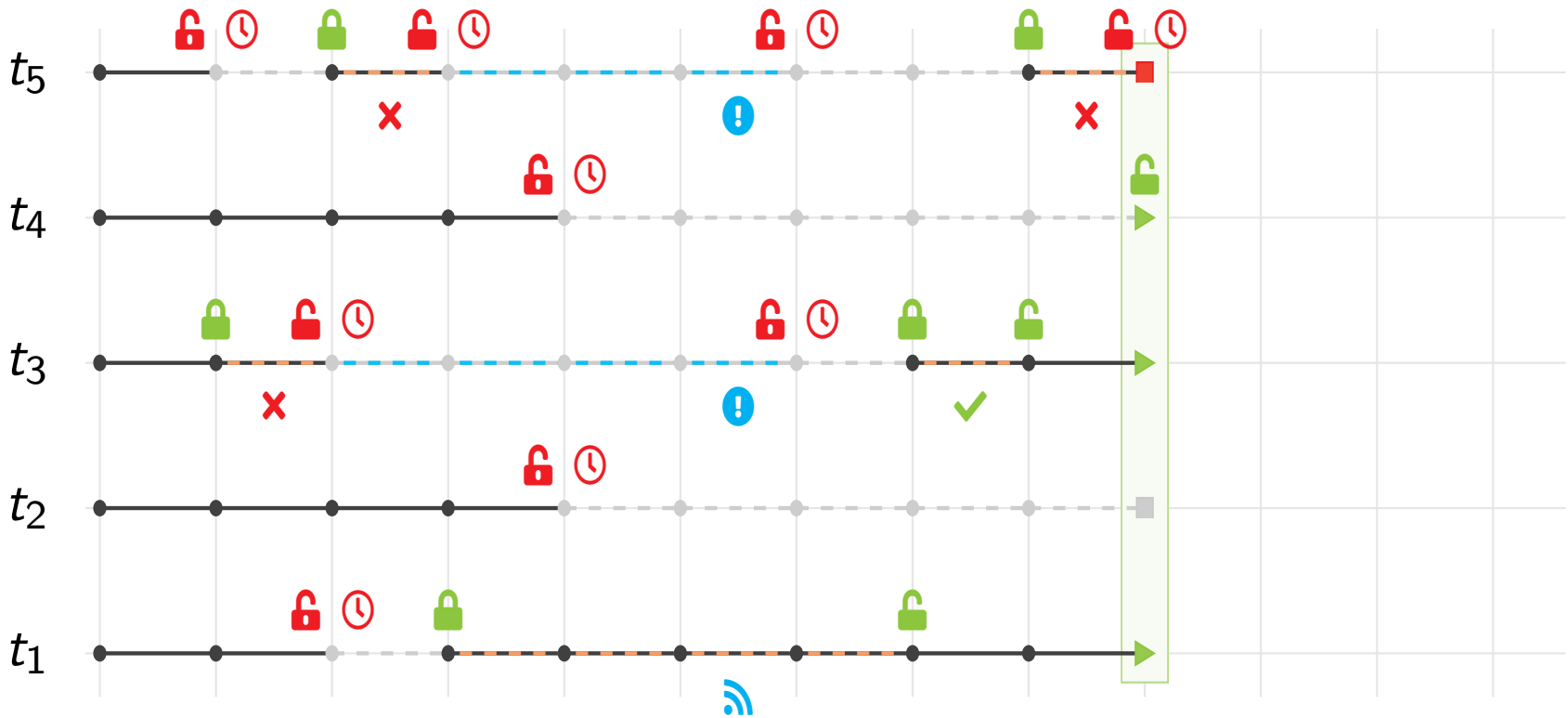
Ukázka 1



- t_5 (c) `pthread_mutex_lock()`, vstupuje do kritické sekce
- t_4 (c) `pthread_mutex_lock()`, čeká
- t_3 (c) podmínka platí $\rightarrow$ `pthread_mutex_unlock()`, opouští kritickou sekci
- t_2 (p) `pthread_mutex_lock()`, čeká
- t_1 (p) běží

Podmínková proměnná

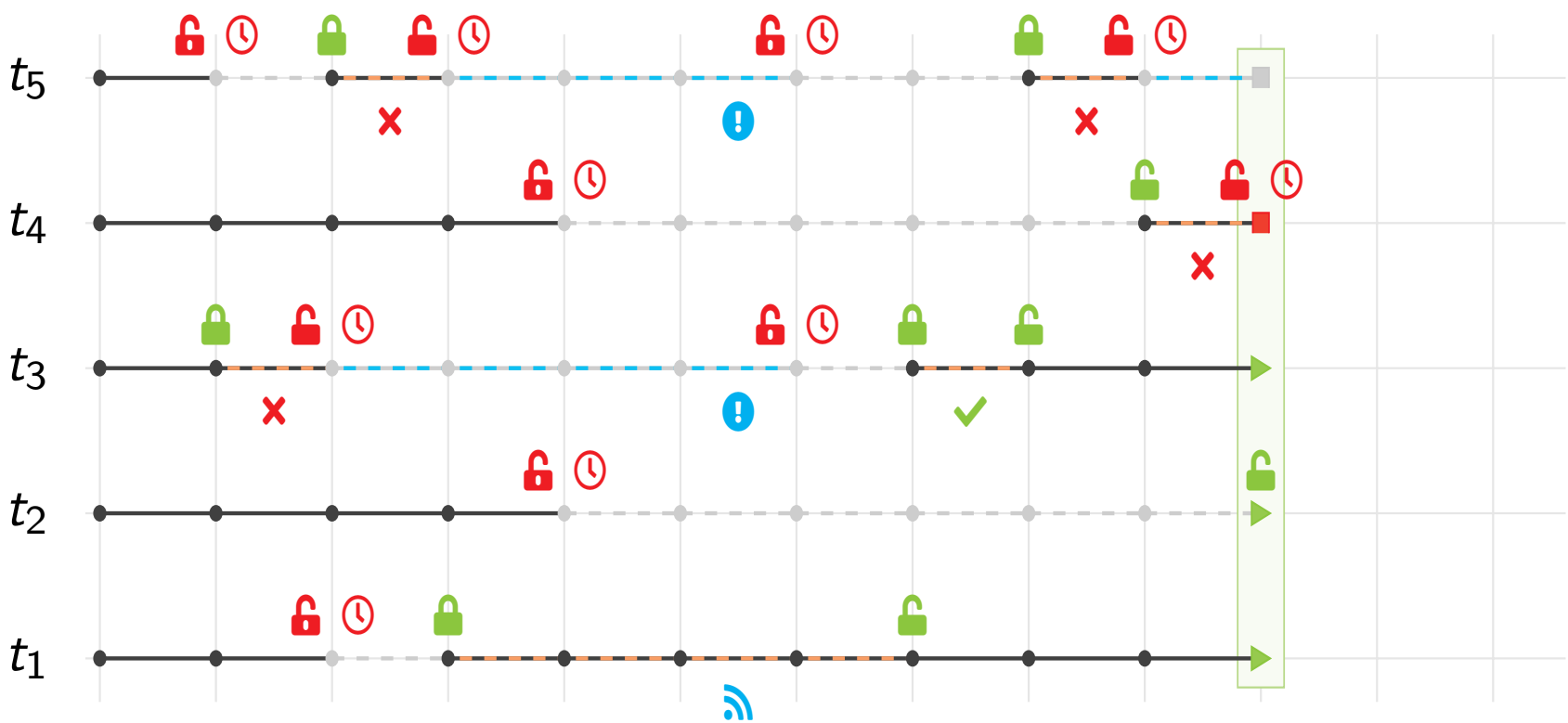
Ukázka 1



- t_5 (c) podmínka neplatí → `pthread_cond_wait()`, uvolní zámek
- t_4 (c) `pthread_mutex_lock()`, vstupuje do kritické sekce
- t_3 (c) běží
- t_2 (p) `pthread_mutex_lock()`, čeká
- t_1 (p) běží

Podmínková proměnná

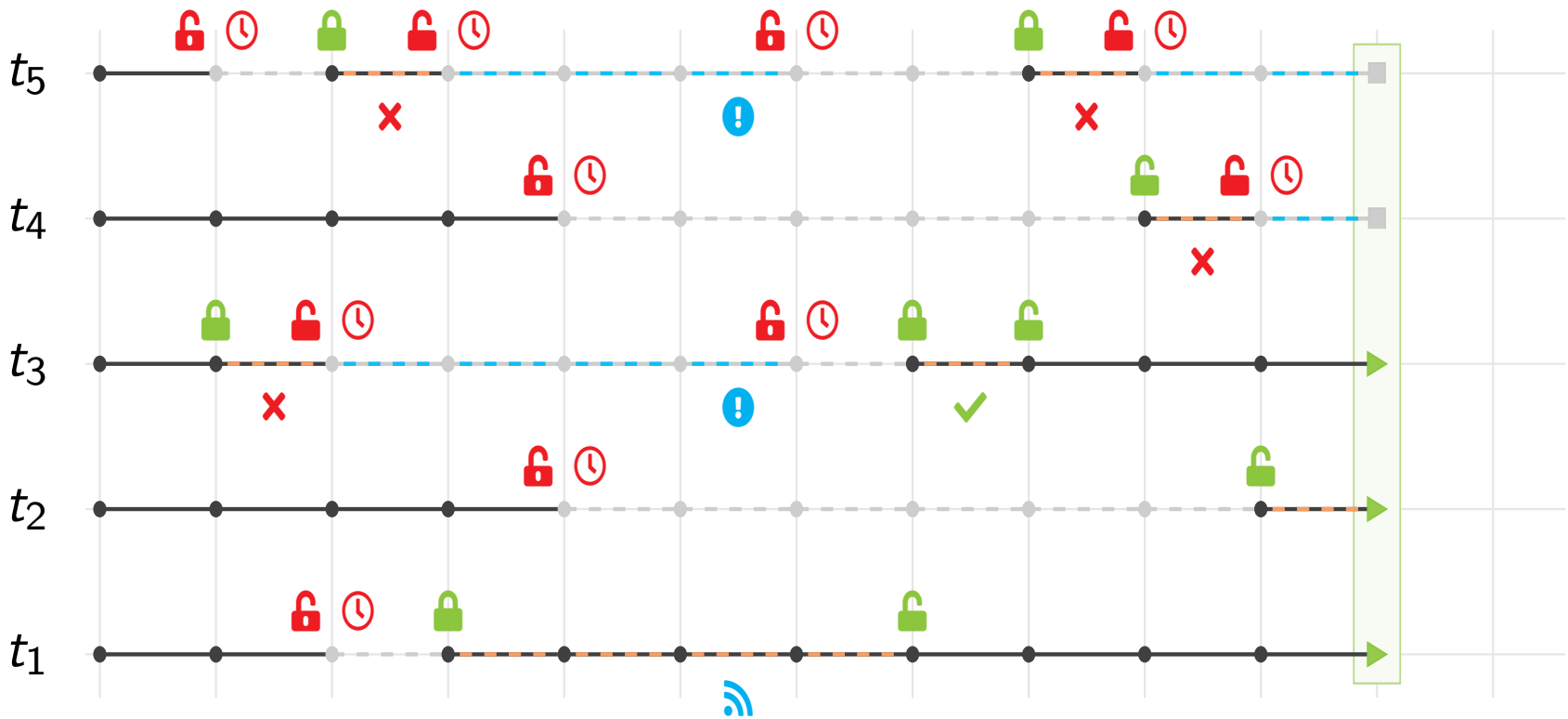
Ukázka 1



- t_5 (c) `pthread_cond_wait()`, čeká
- t_4 (c) podmínka neplatí → `pthread_cond_wait()`, uvolní zámek
- t_3 (c) běží
- t_2 (p) `pthread_mutex_lock()`, vstupuje do kritické sekce
- t_1 (p) běží

Podmínková proměnná

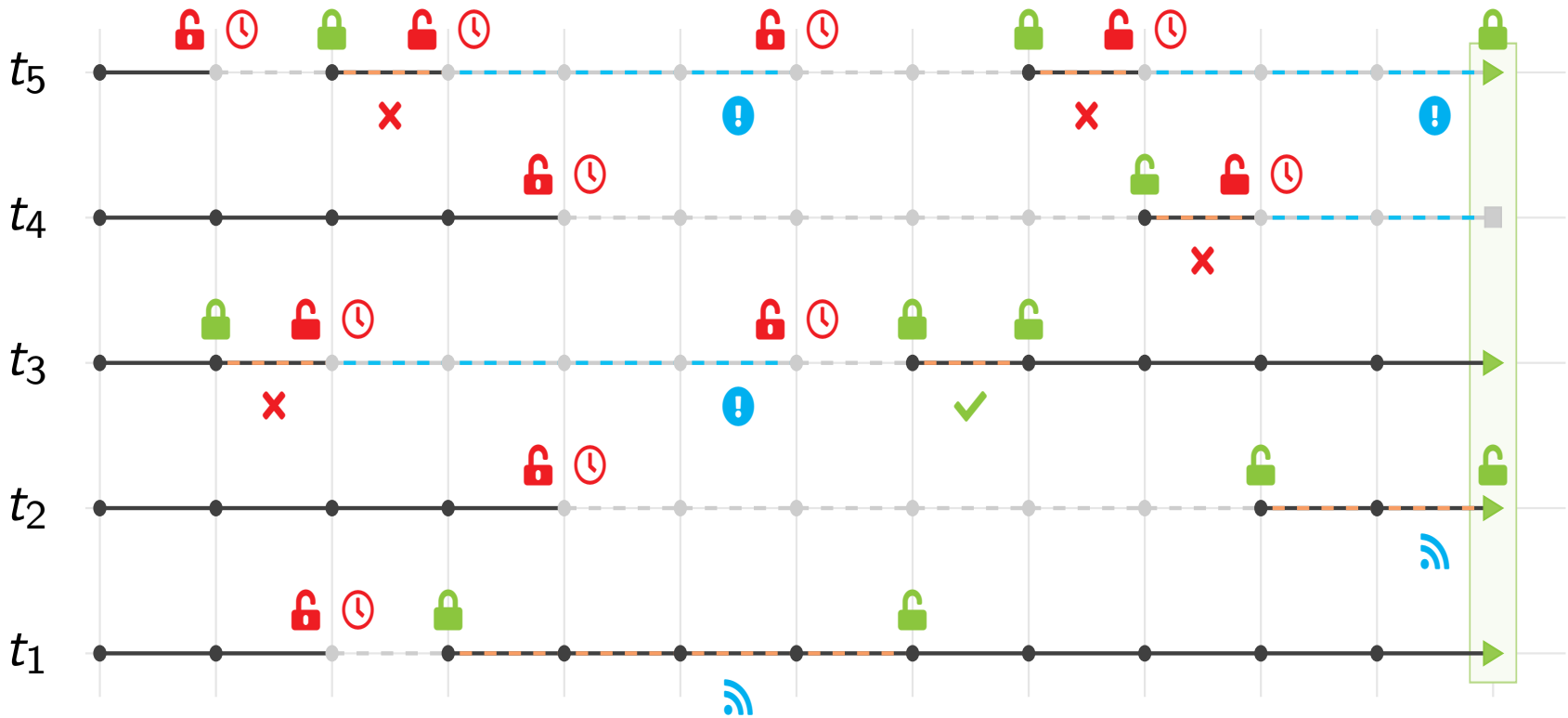
Ukázka 1



- t_5 (c) `pthread_cond_wait()`, čeká
- t_4 (c) `pthread_cond_wait()`, čeká
- t_3 (c) běží
- t_2 (p) běží v kritické sekci (vkládá prvek)
- t_1 (p) běží

Podmínková proměnná

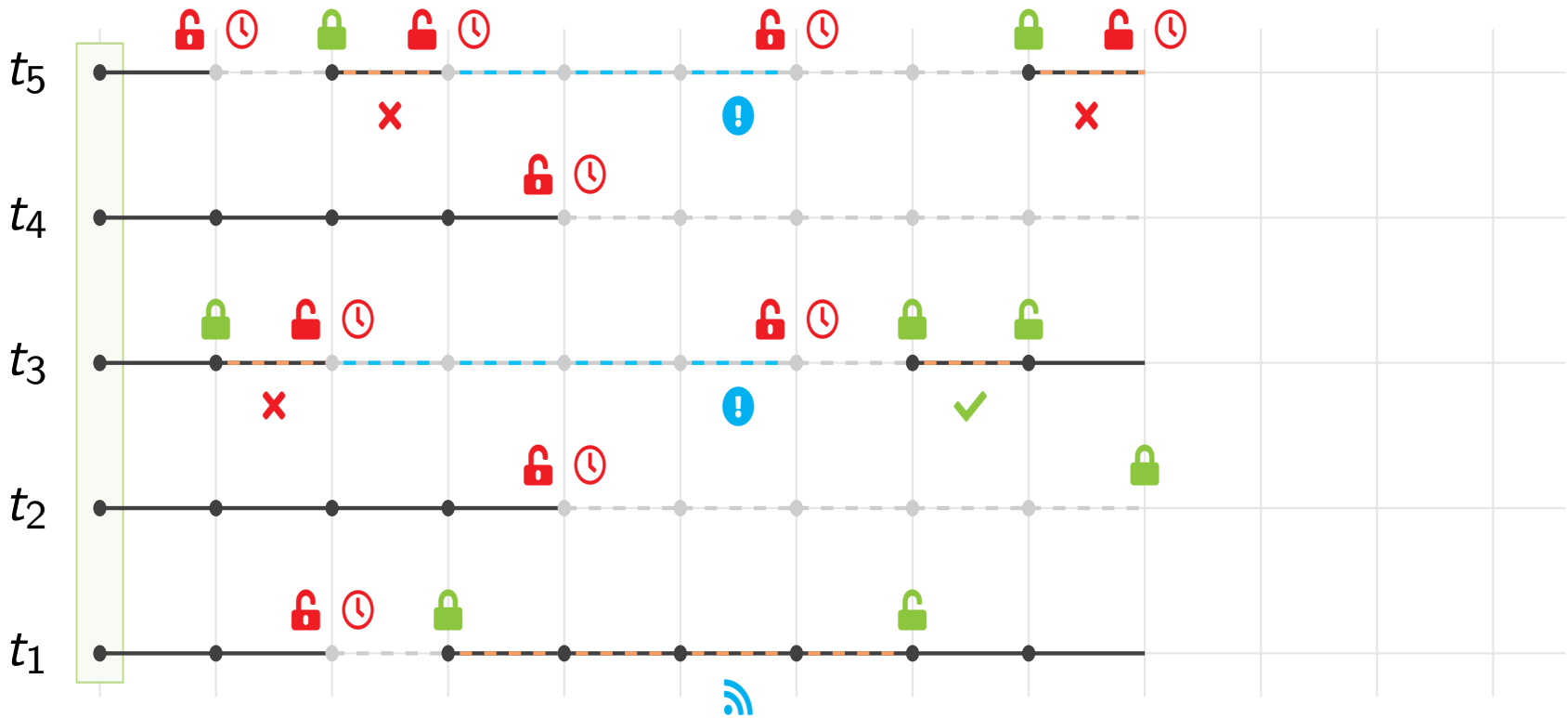
Ukázka 1



- t_5 (c) `pthread_cond_wait()` → `pthread_mutex_lock()`, vstupuje do kritické sekce
- t_4 (c) `pthread_cond_wait()`, čeká
- t_3 (c) běží
- t_2 (p) `pthread_cond_signal()`, `pthread_mutex_unlock()`, opouští kritickou sekci
- t_1 (p) běží

Podmínková proměnná

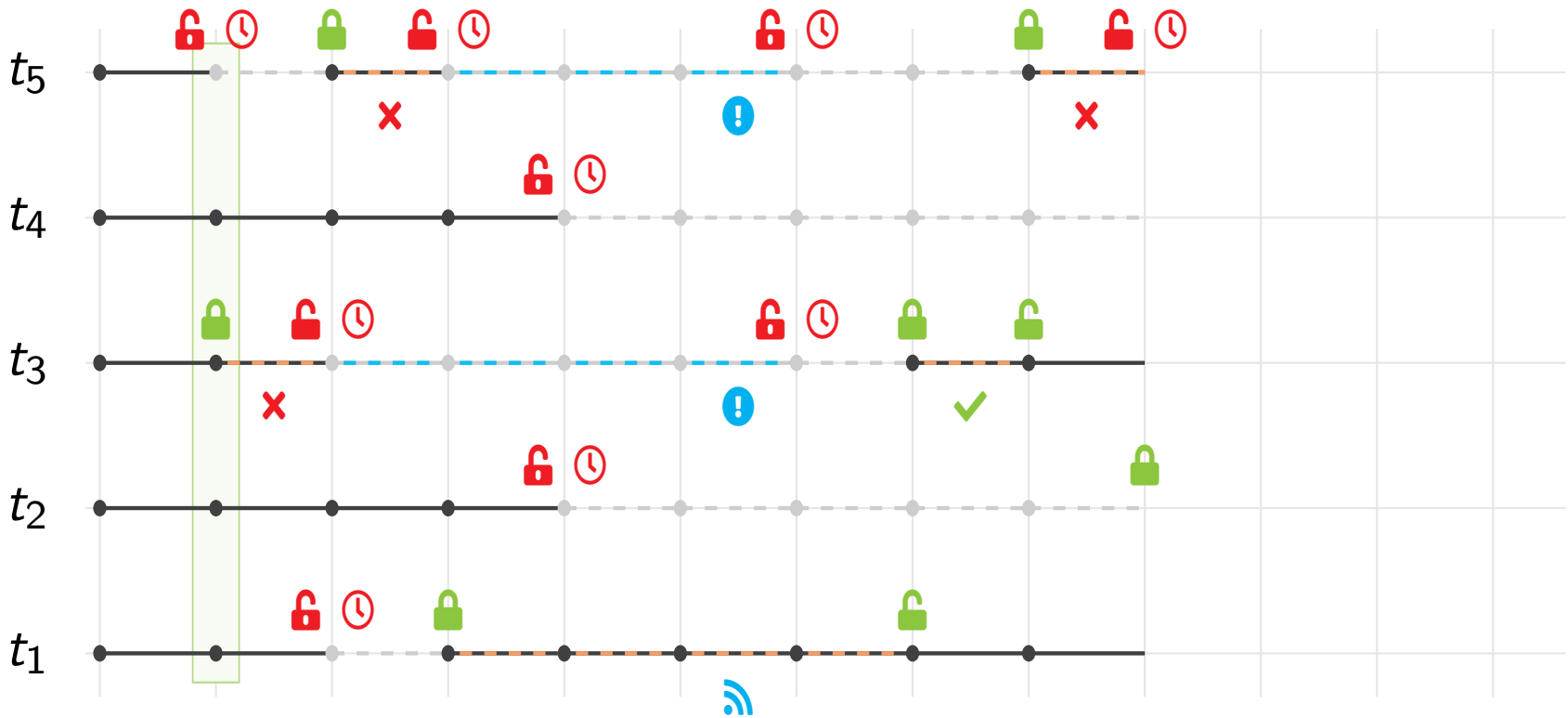
Ukázka 2



- t_5 (c) ...
- t_4 (c) ...
- t_3 (c) ...
- t_2 (p) ...
- t_1 (p) ...

Podmínková proměnná

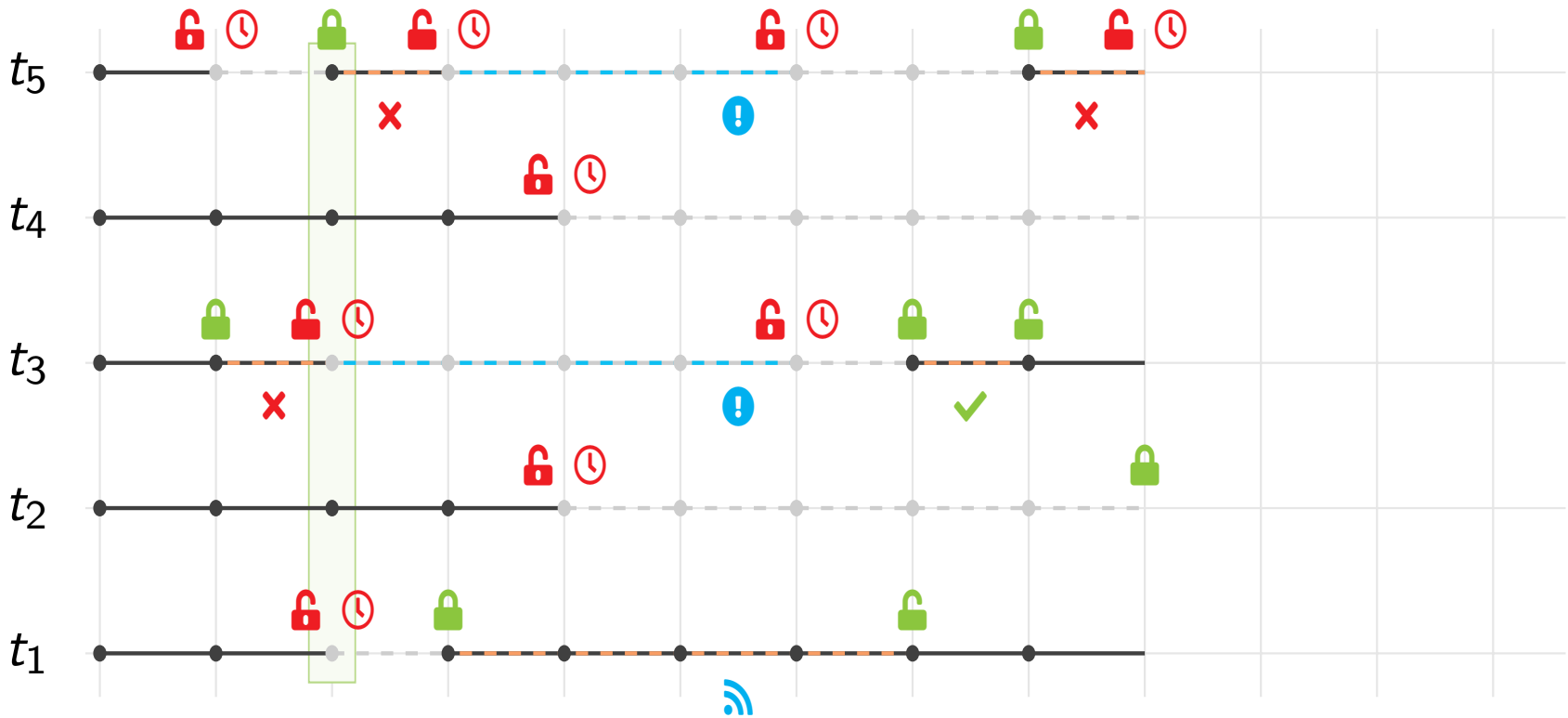
Ukázka 2



- t_5 (c) ...
- t_4 (c) ...
- t_3 (c) ...
- t_2 (p) ...
- t_1 (p) ...

Podmínková proměnná

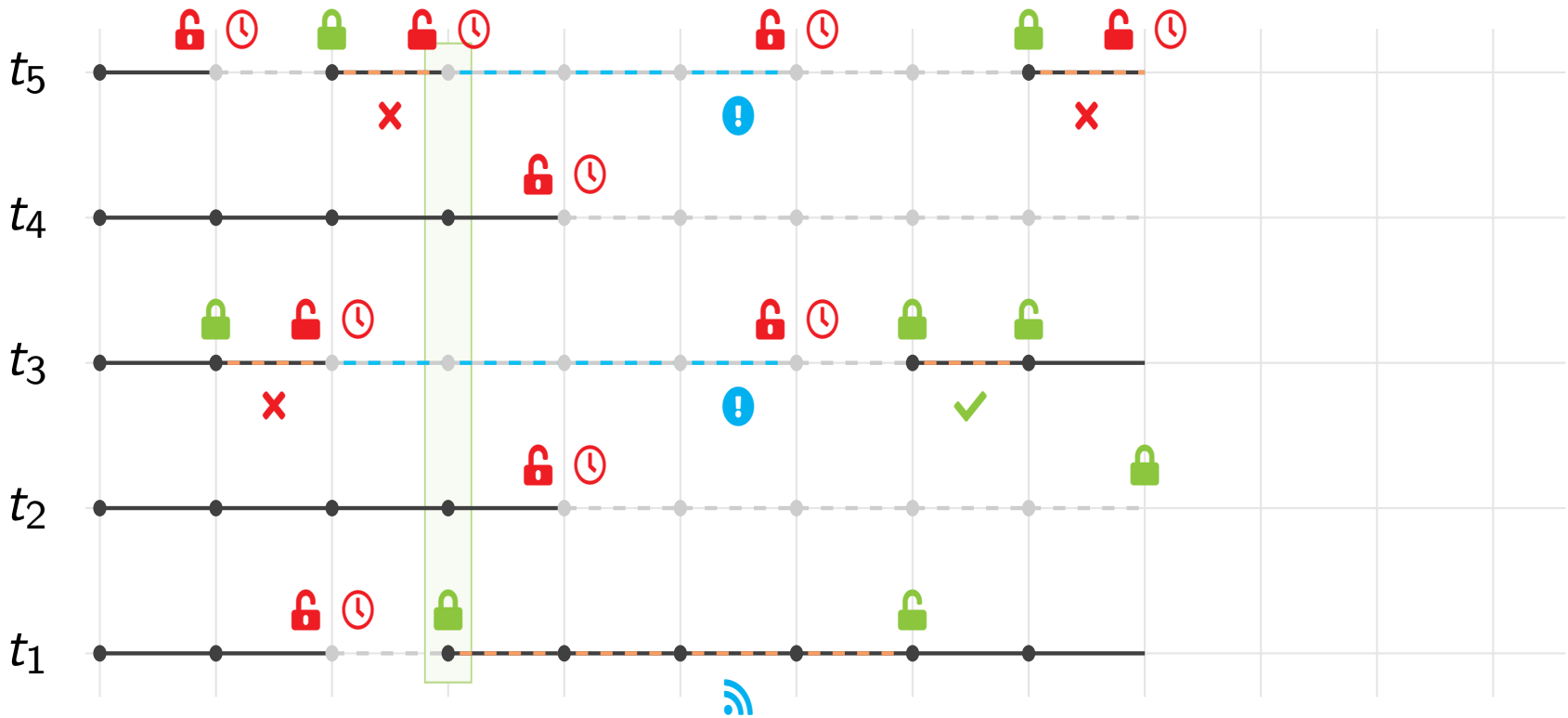
Ukázka 2



- t_5 (c) ...
- t_4 (c) ...
- t_3 (c) ...
- t_2 (p) ...
- t_1 (p) ...

Podmínková proměnná

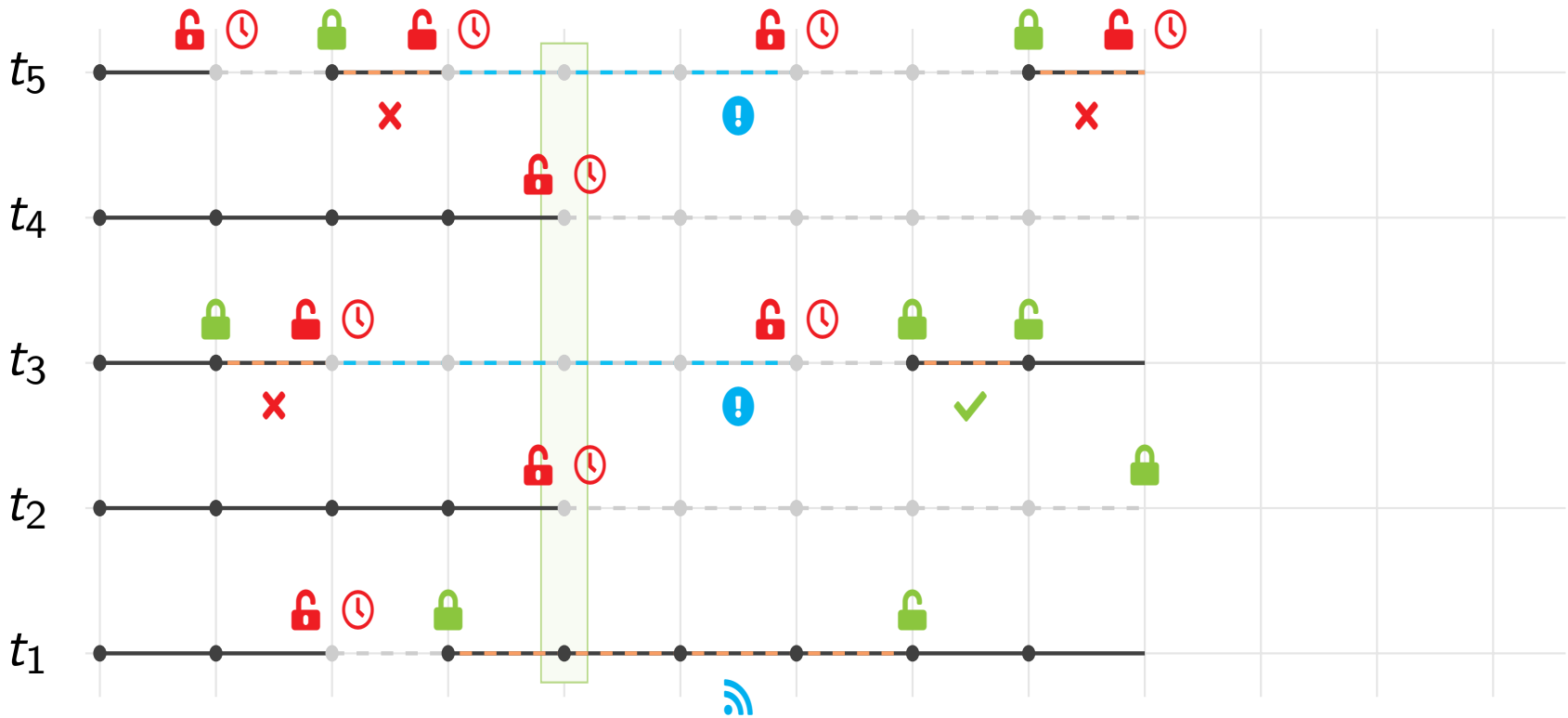
Ukázka 2



- t_5 (c) ...
- t_4 (c) ...
- t_3 (c) ...
- t_2 (p) ...
- t_1 (p) ...

Podmínková proměnná

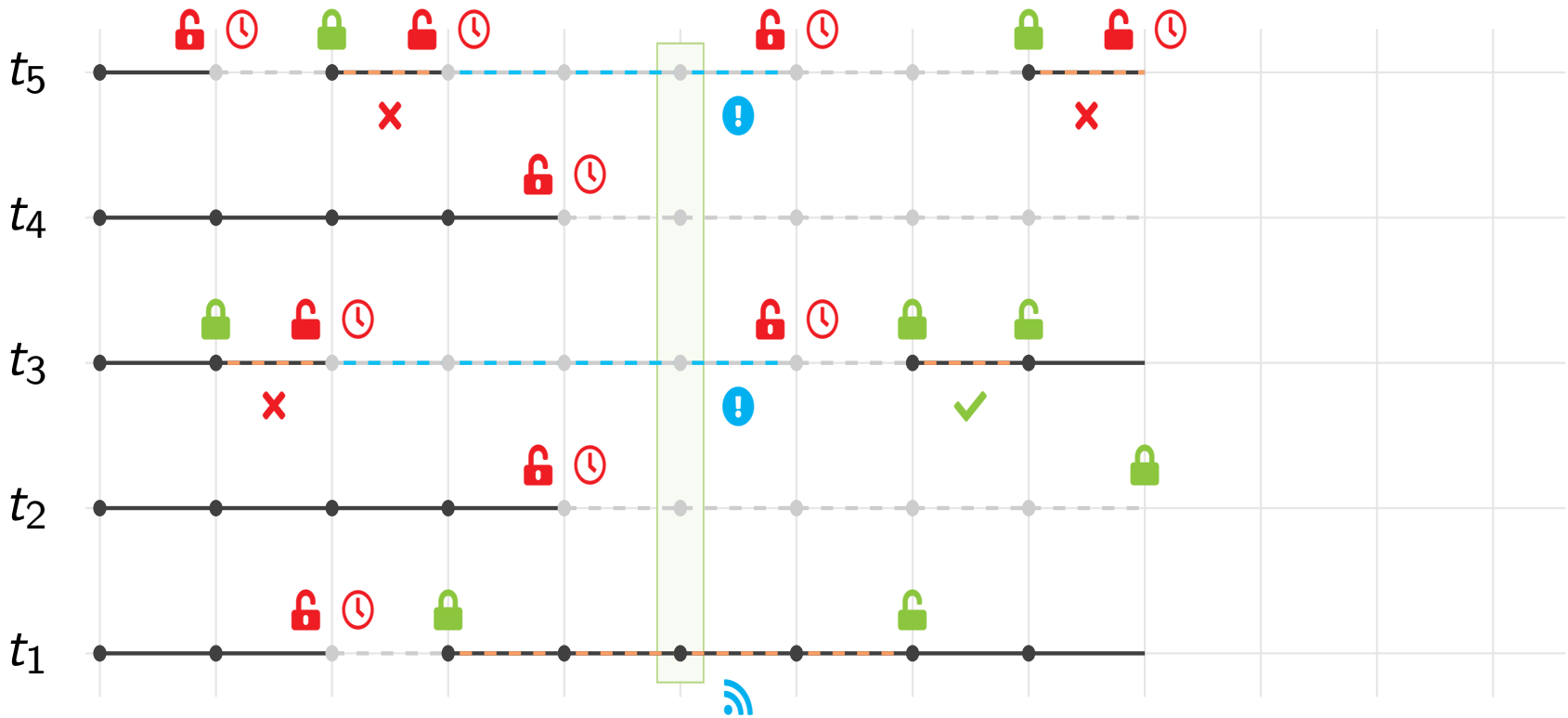
Ukázka 2



- t_5 (c) ...
- t_4 (c) ...
- t_3 (c) ...
- t_2 (p) ...
- t_1 (p) ...

Podmínková proměnná

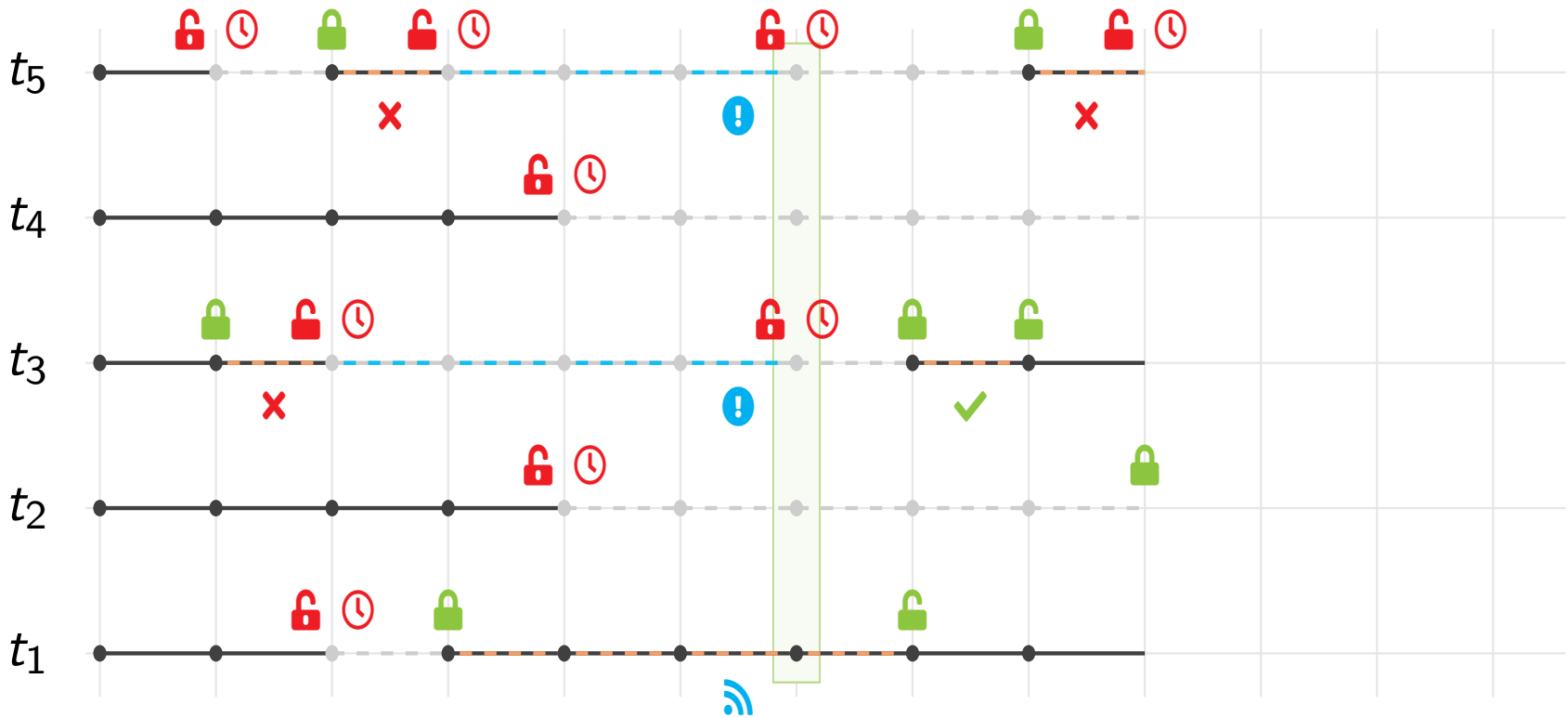
Ukázka 2



- t_5 (c) ...
- t_4 (c) ...
- t_3 (c) ...
- t_2 (p) ...
- t_1 (p) ...

Podmínková proměnná

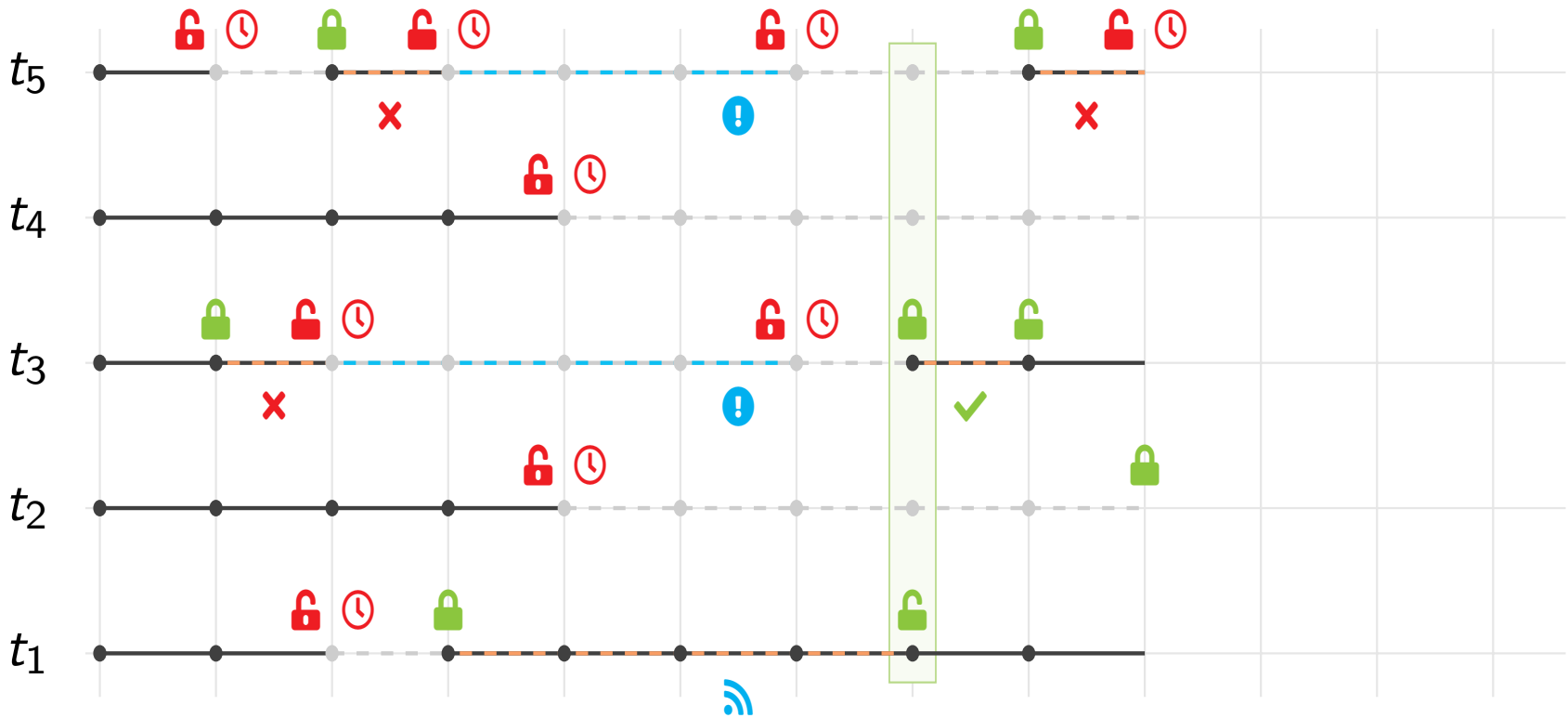
Ukázka 2



- t_5 (c) ...
- t_4 (c) ...
- t_3 (c) ...
- t_2 (p) ...
- t_1 (p) ...

Podmínková proměnná

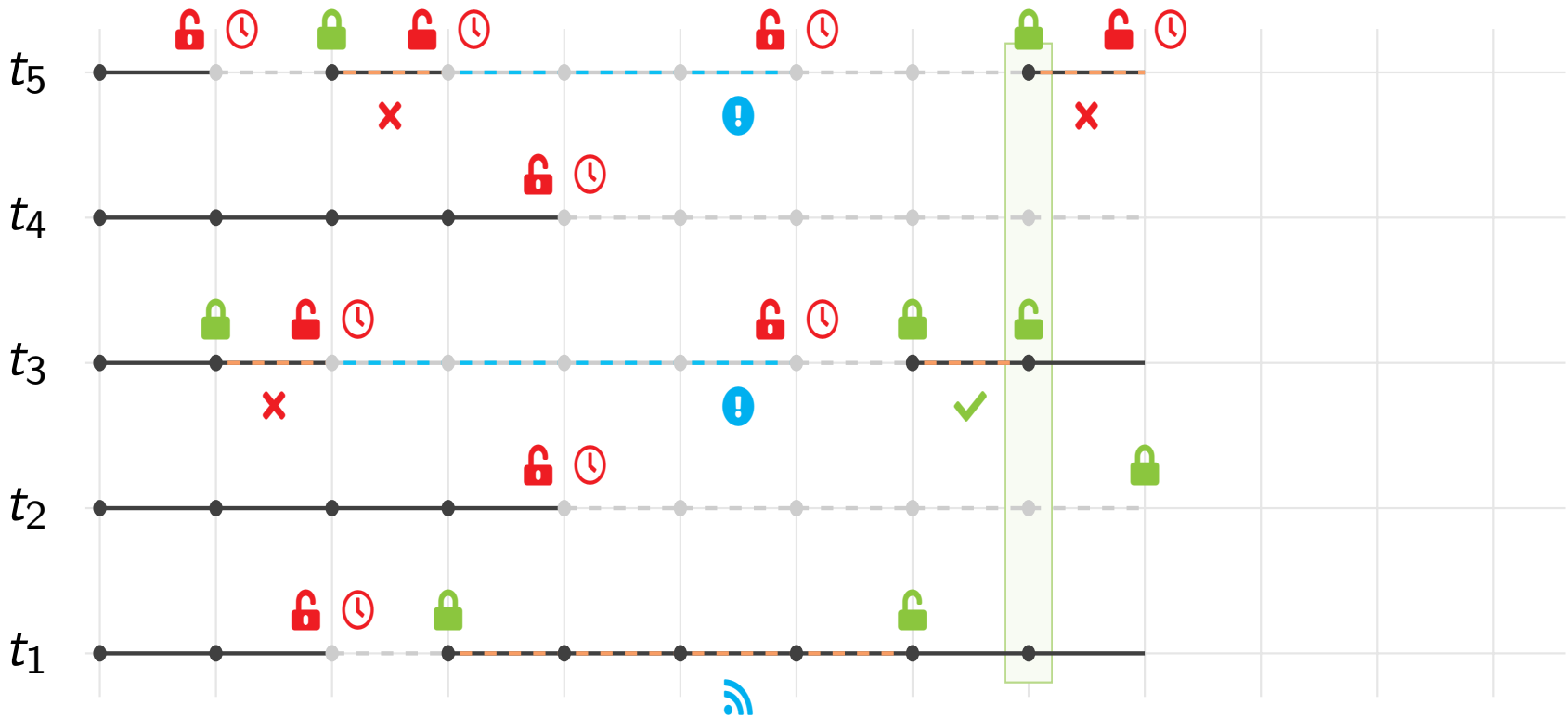
Ukázka 2



- t_5 (c) ...
- t_4 (c) ...
- t_3 (c) ...
- t_2 (p) ...
- t_1 (p) ...

Podmínková proměnná

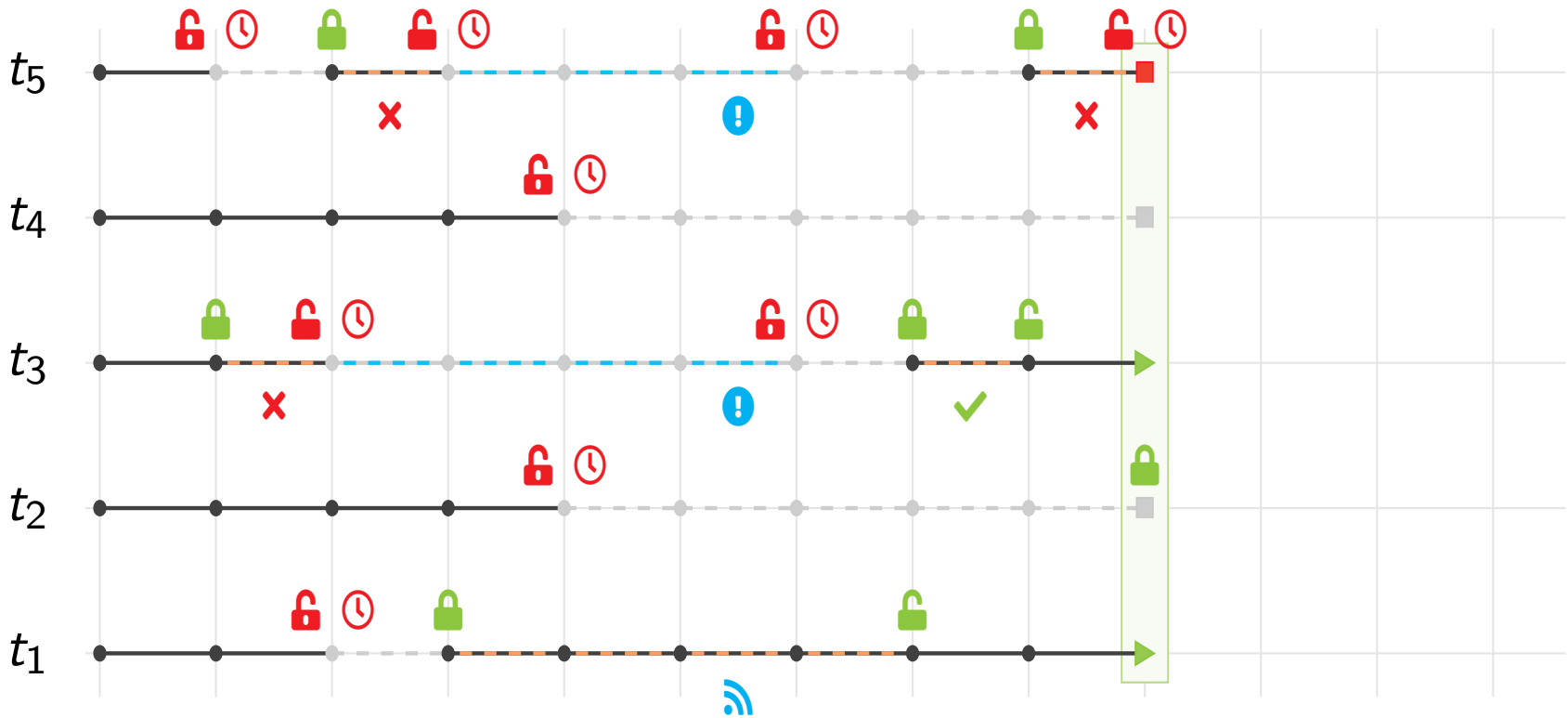
Ukázka 2



- t_5 (c) ...
- t_4 (c) ...
- t_3 (c) ...
- t_2 (p) ...
- t_1 (p) ...

Podmínková proměnná

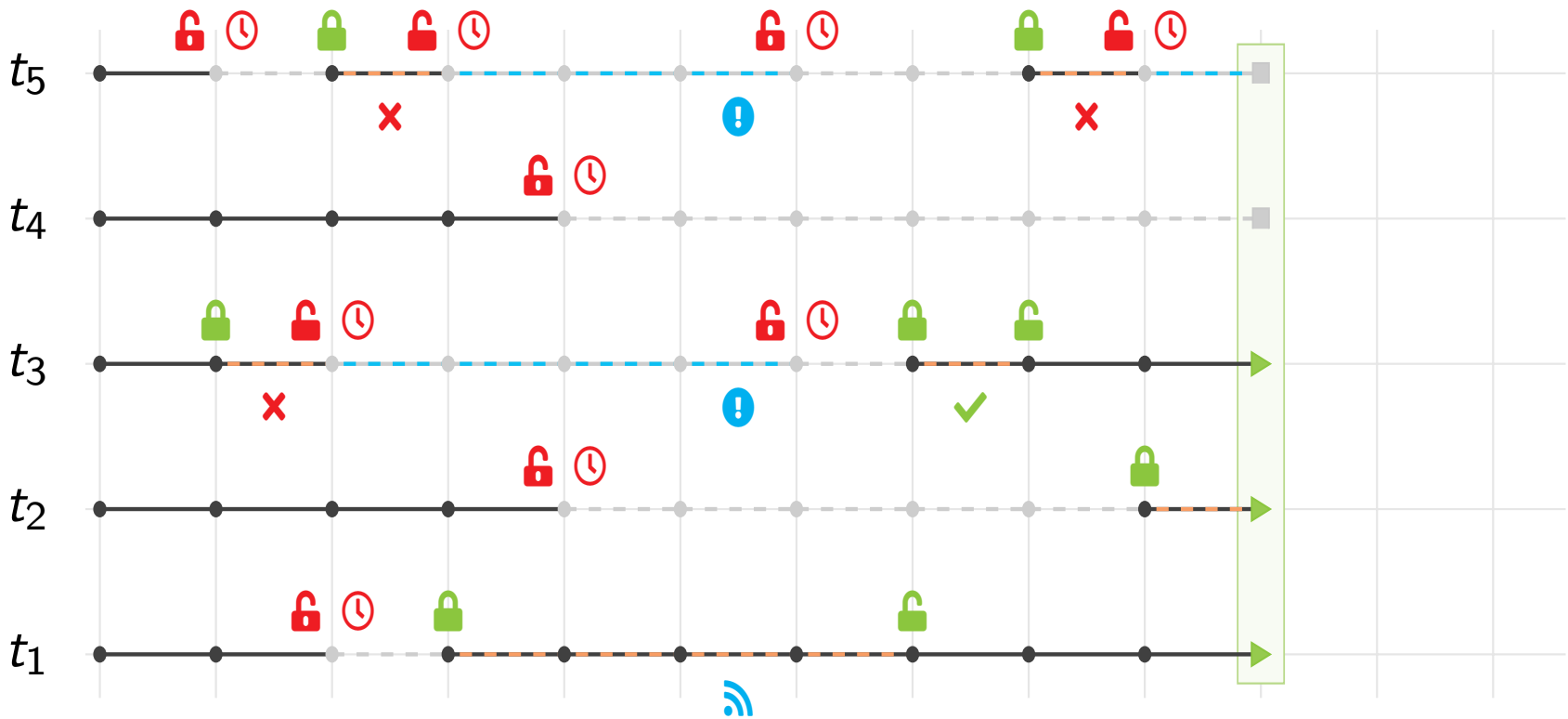
Ukázka 2



- t_5 (c) podmínka neplatí → `pthread_cond_wait()`, uvolní zámek
- t_4 (c) `pthread_mutex_lock()`, čeká
- t_3 (c) běží
- t_2 (p) `pthread_mutex_lock()`, vstupuje do kritické sekce
- t_1 (p) běží

Podmínková proměnná

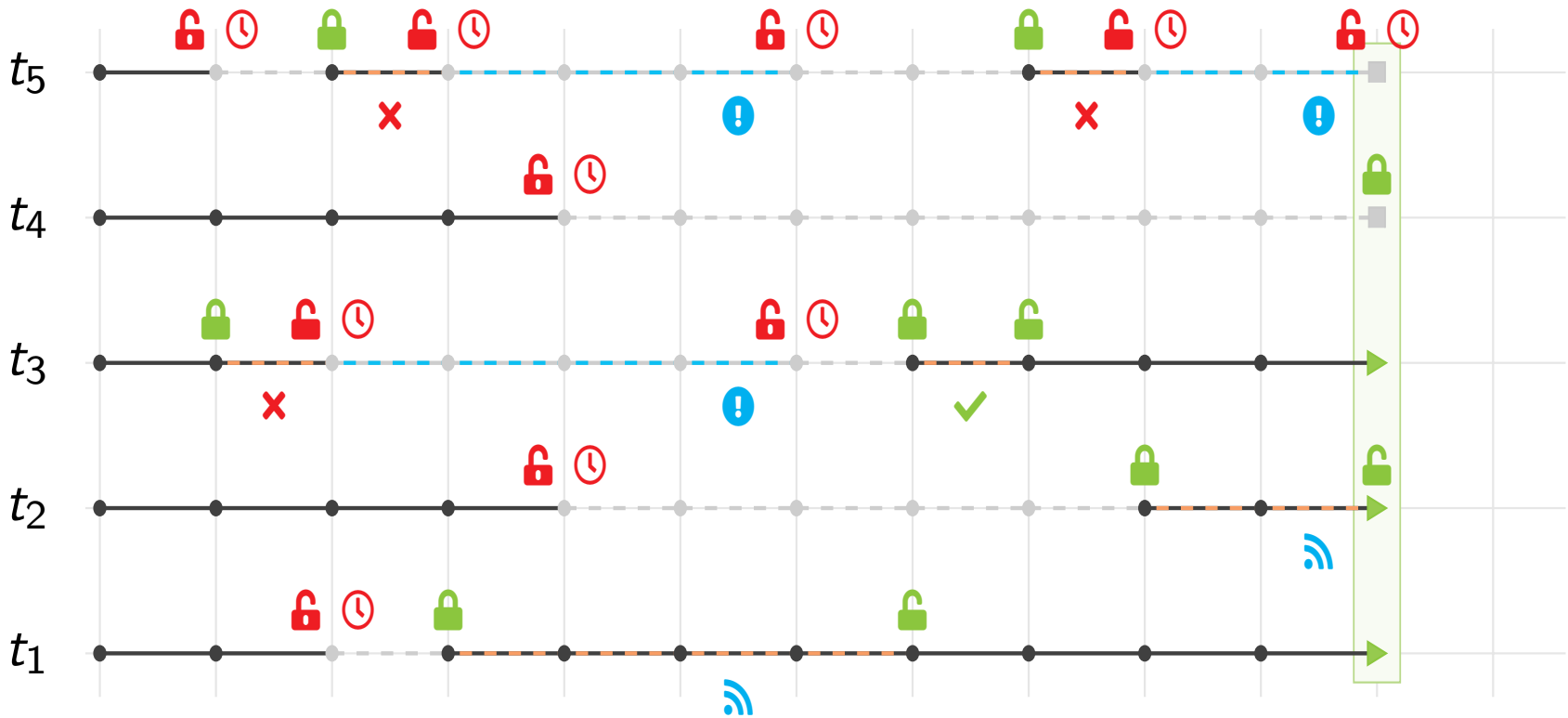
Ukázka 2



- t_5 (c) `pthread_cond_wait()`, čeká
- t_4 (c) `pthread_mutex_lock()`, čeká
- t_3 (c) běží
- t_2 (p) běží v kritické sekci (vkládá prvek)
- t_1 (p) běží

Podmínková proměnná

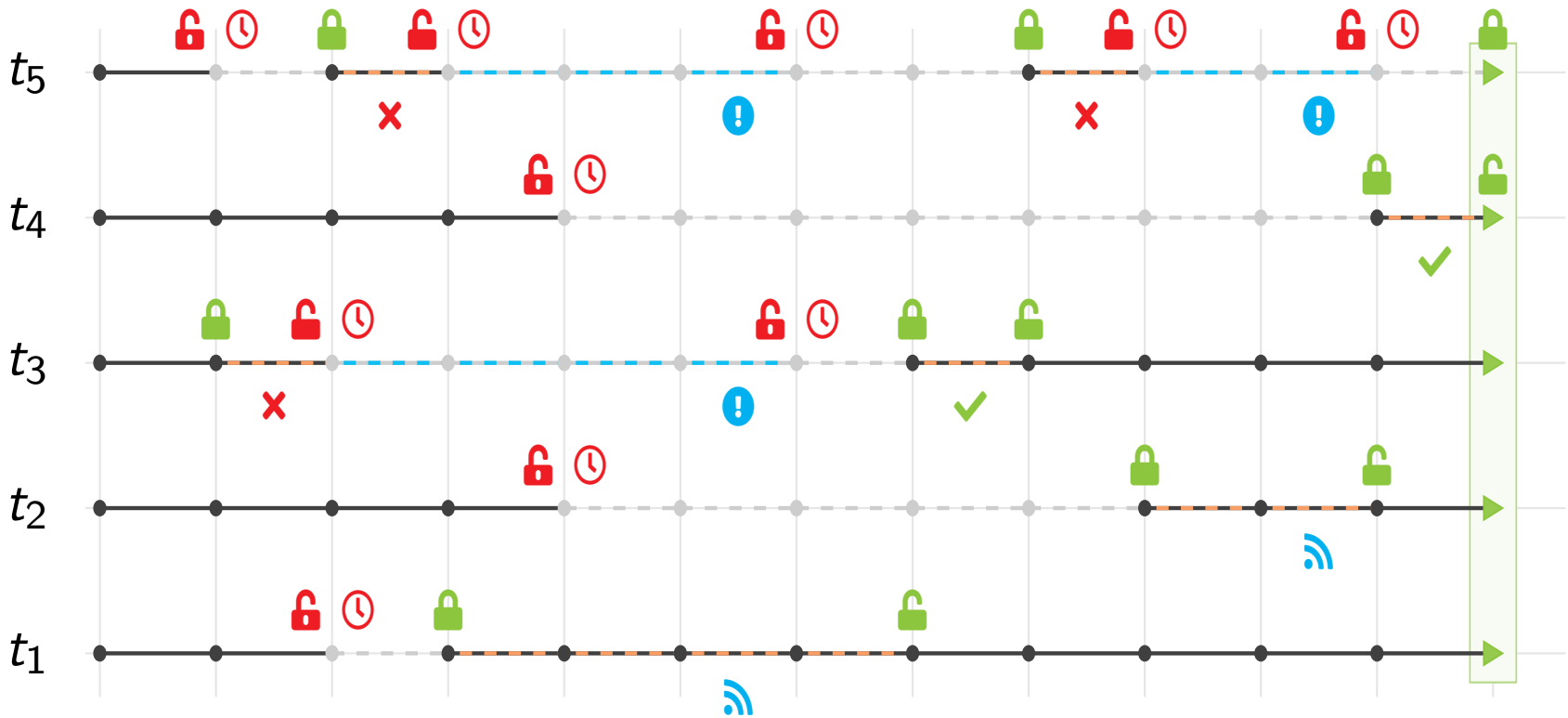
Ukázka 2



- t_5 (c) `pthread_cond_wait()` → `pthread_mutex_lock()`, uspí se
- t_4 (c) `pthread_mutex_lock()`, vstupuje do kritické sekce
- t_3 (c) běží
- t_2 (p) `pthread_cond_signal()`, `pthread_mutex_unlock()`, opouští kritickou sekci
- t_1 (p) běží

Podmínková proměnná

Ukázka 2



- t_5 (c) `pthread_mutex_lock()`, probudí se
- t_4 (c) `podmínka platí` → `pthread_mutex_unlock()`, opouští kritickou sekci
- t_3 (c) běží
- t_2 (p) běží
- t_1 (p) běží