Unstable-Funcons-beta: Synchronising *

The PLanCompS Project

`Synchronising.cbs` | PLAIN | PRETTY

OUTLINE

Thread synchronisation

[Syncs
Datatype syncs
Funcon sync-create
Funcon sync-feature
Funcon is-sync-feature
Sync features
Datatype sync-features
Funcon sync-waiting-list
Funcon sync-held
Funcon sync-holder
Funcon sync-count
Funcon sync-feature-create]

Thread synchronisation can be supported in many different ways: semaphores, exclusive and shared locks, conditions, barriers, rendezvous, spin-locks, etc. They generally involve the execution of one or more threads being blocked while they wait for some synchronisation request to be granted by a synchroniser due to a step by some unblocked thread. Blocking may involve thread suspension or repeated requests.

In general, the effect of granting a sync needs to be atomic, to preclude preemption. However, the execution of the thread that caused the request to be granted might continue without yielding, thereby delaying the resumed execution of the requesting thread. Synchronisation ensures that the executions of two or more threads are at particular points at the same time, but it does *not* require their next steps to be simultaneous.

Syncs are mutable structures that map sync features to variables (some fields may be constant values). Inspecting and updating sync features should be atomic, in case threads are preemptible.

Syncs A sync is formed from its features:

Datatype syncs ::= sync(_ : sync-feature-maps)

sync-create(M^+) checks that the specified features are distinct. (It could also check required feature constraints.)

*Suggestions for improvement: `plancomps@gmail.com`.
Reports of issues: `https://github.com/plancomps/CBS-beta/issues`.

Funcon sync-create(M^+ : sync-feature-maps$^+$) : ⇒ syncs
⇝ sync checked map-unite M^+

sync-feature(SY, SF) selects the feature SF from SY:

Funcon sync-feature(_ : syncs, _ : sync-features) : ⇒ values
Rule sync-feature(sync(SFM : sync-feature-maps), SF : sync-features) ⇝
checked map-lookup(SFM, SF)

is-sync-feature(SY, SF) tests whether SY has the feature SF:

Funcon is-sync-feature(_ : syncs, _ : sync-features) : ⇒ values
Rule is-sync-feature(sync(SFM : sync-feature-maps), SF : sync-features) ⇝
is-in-set(SF, dom SFM)

Sync features Combinations of the following features support various kinds of locks and notifications.

Datatype sync-features ::= sync-waiting-list
| sync-held
| sync-holder
| sync-count

Auxiliary Type sync-feature-maps ⇝ maps(sync-features, values)

A field for each feature is created independently:

Funcon sync-feature-create(_ : sync-features) : ⇒ sync-feature-maps

sync-waiting-list stores pending requests in the order of receipt, together with the requesting thread-ids:

Rule sync-feature-create sync-waiting-list ⇝
{sync-waiting-list ↦
allocate-initialised-variable(lists(values), [ ])}

sync-held stores whether a lock is currently held:

Rule sync-feature-create sync-held ⇝
{sync-held ↦
allocate-initialised-variable(booleans, false)}

sync-holder stores the current holder of a lock, if any:

Rule sync-feature-create sync-holder ⇝
{sync-holder ↦
allocate-variable(thread-ids)}

sync-count stores a counter. Different kinds of locks and notifications use the counter in different ways, e.g., shared locks use it for the number of threads currently holding the lock:

Rule sync-feature-create sync-count ⇝
{sync-count ↦
allocate-initialised-variable(nats, 0)}

sync-waiting-list-add(SY, V) adds V to the waiting-list of SY:

Auxiliary Funcon sync-waiting-list-add(SY : syncs, V : values) : ⇒ null-type
⇝ assign(
sync-feature(SY, sync-waiting-list),
list-append(assigned sync-feature(SY, sync-waiting-list), [V]))

sync-waiting-list-head-remove(SY) removes the first value from the waiting-list of SY:

Auxiliary Funcon sync-waiting-list-head-remove(SY : syncs) : ⇒ values
⇝ give(
checked list-head assigned sync-feature(SY, sync-waiting-list),
sequential(
assign(
sync-feature(SY, sync-waiting-list),
checked list-tail assigned sync-feature(SY, sync-waiting-list)),
given))

Various kinds of locks and notifications are represented by sync feature maps, together with funcons that (atomically) inspect and update them accordngly.