The CA Action Concept

Department of Computing Science
University of Newcastle upon Tyne

Outline

- ☞Basic Terms: Objects, Threads and Actions
- ☞Inter-Thread Concurrency
- ☞External Behaviour of a CA Action
- ☞Internal Activities of a CA Action
- ☞Recursive View
- ☞Error Recovery & Fault Tolerance
- ☞Towards Formalization of the CA Action Concept
- ☞Conclusions

Elements

Objects

An object is a named entity that combines a data structure (internal state) with its associated operations that determines the externally visible behaviour of the object.

Threads

Threads are the agents of computation (in a concurrent system). A thread is an active entity that is responsible for executing a *sequence* of operations on objects.
(Threads can exist syntactically, e.g. in Java, or as a purely run-time concept.)

Concurrency

☞**Kinds of Inter-Thread Concurrency:**

Independent Concurrency

No sharing or interacting each other (e.g. disjoint object sets)

Competitive Concurrency

Sharing but no (explicit) cooperating (e.g. sharing not ordered)

Cooperative Concurrency

Cooperating & interacting each other to achieve a joint goal (i.e. each thread is responsible only for a part of the joint goal.)

Thread Cooperation

☞ How concurrent execution threads cooperate? i.e. how different threads can be glued together?

☞ We attempt to model inter-thread cooperation as information transfer via shared objects. Such an abstraction may cover actual forms of inter-thread cooperation such as:

Inter-thread Communication
1) updating shared objects (plus some synchro. mechanism)
2) message passing (without requiring shared storage)

Inter-thread Synchronization
1) condition synchronization (usually no data passed)
2) exclusion synchronization (usually for shared object schemes)

Actions

Actions

An action is an abstraction that allows the application programmer to group a *set* of operations on objects into a logical execution unit.
(An action may be associated with desirable properties, e.g. ACID.)

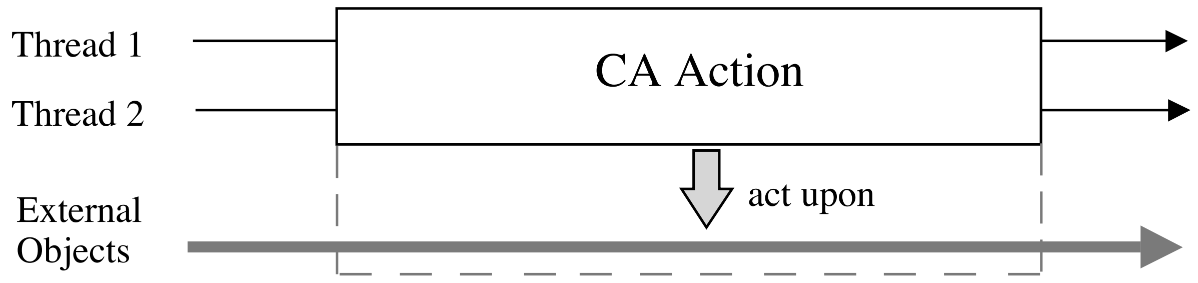
It provides another way of gluing multiple execution threads together.

A CA action is an enclosure of recoverable activities of multiple cooperating threads.

External Behaviour of a CA Action

1) Participating threads enter the action (logically) synchronously
2) Participating threads leave the action (logically) synchronously
3) The action possesses ACID properties wrt external objects

(Failure) Atomicity -- All or nothing*;
Consistency -- If the state was consistent before the action it should be so after;
Isolation -- No interaction between the action and others (actions and threads)*;
Durability -- Any changes made during the action become permanent upon exit.



Coordination

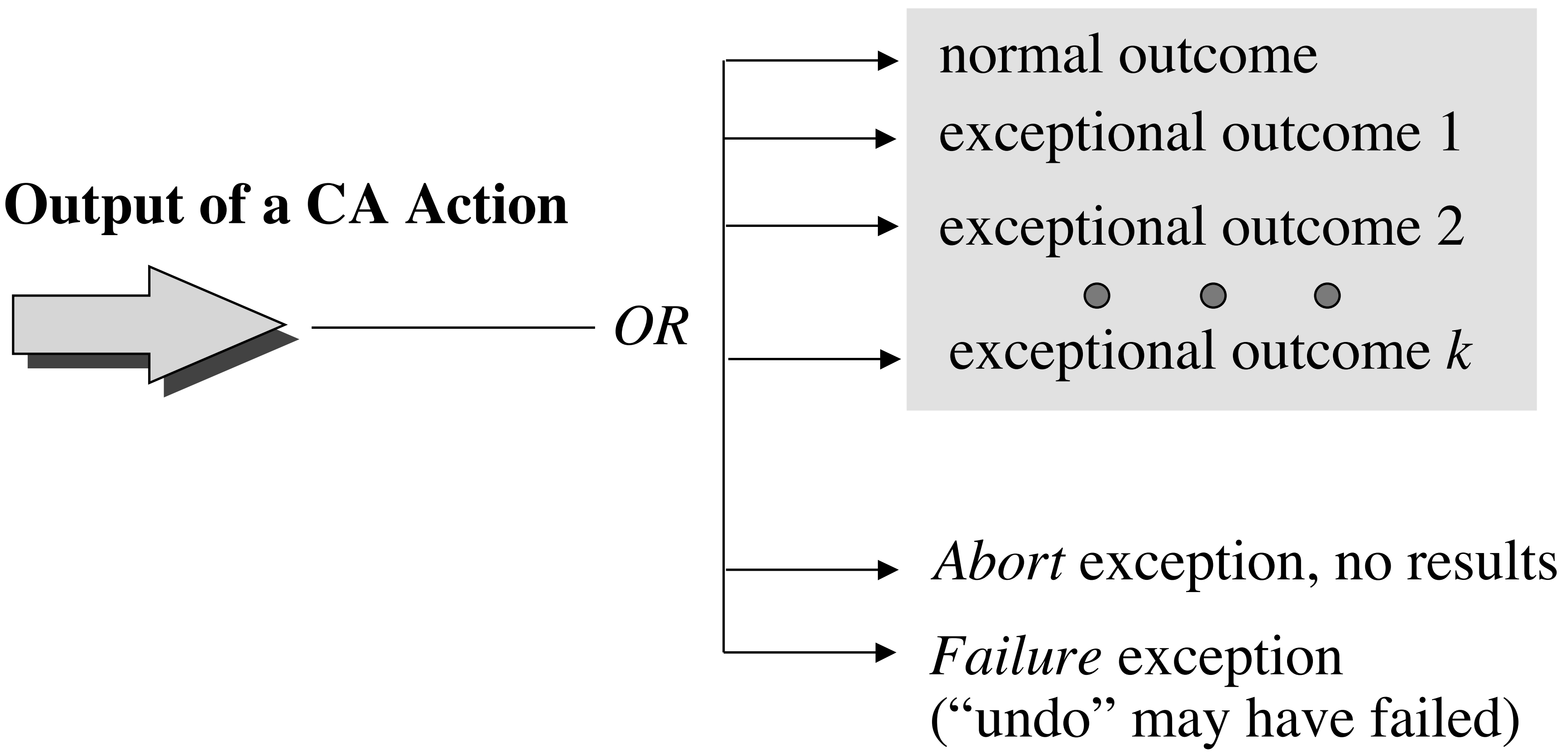
Coordination is needed since a CA Action is concerned in multiple concurrent execution threads (or multi-parties).

1) Coordination upon entrance;

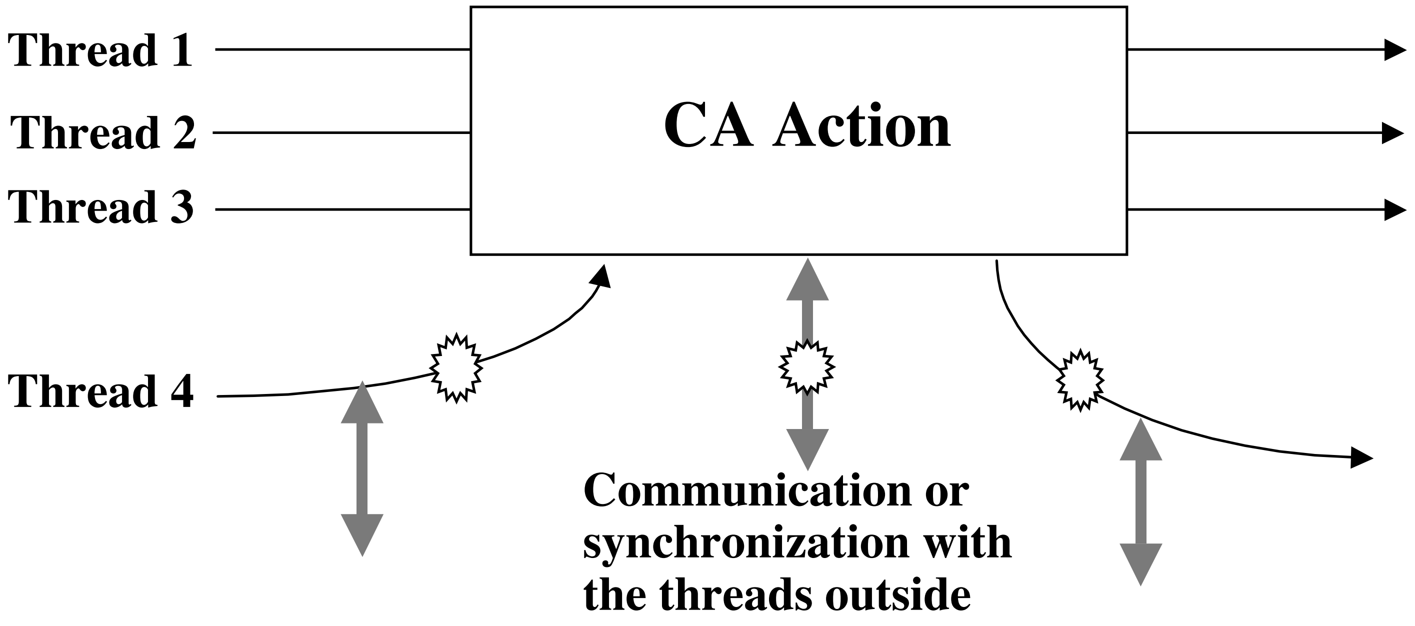
2) Coordination upon exit; and

3) Coordination of handling exceptional situations.

Failure Atomicity

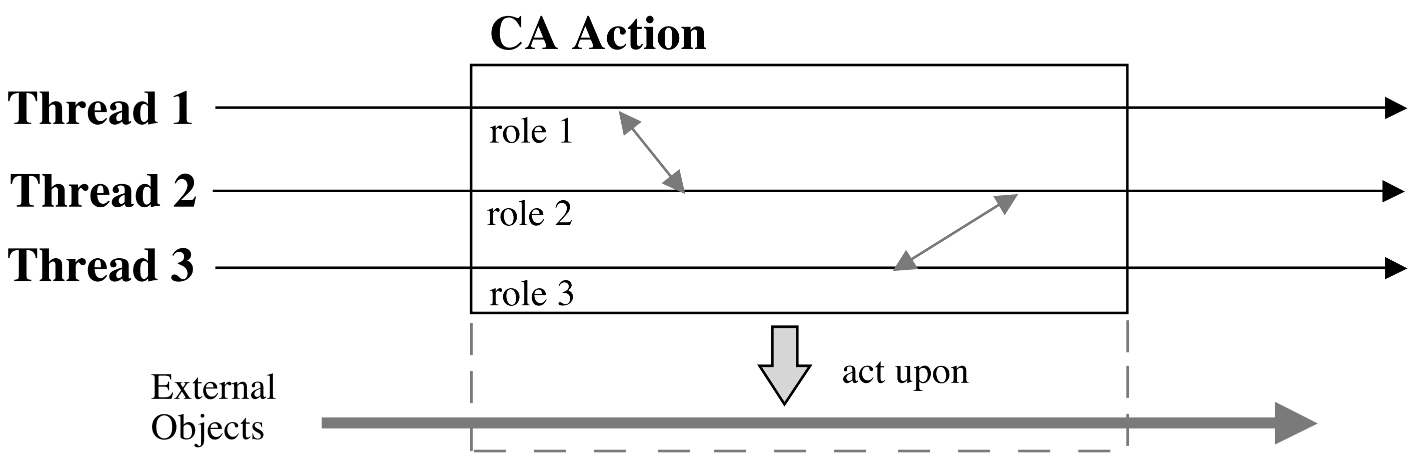


Isolation



1) Direct or indirect interaction with the world outside the CA Action is NOT allowed
2) The possibility of implicit interaction with the outside via external shared objects is ruled out by the atomicity property, i.e. no intermediate results can be seen

Internal Activities of a CA Action

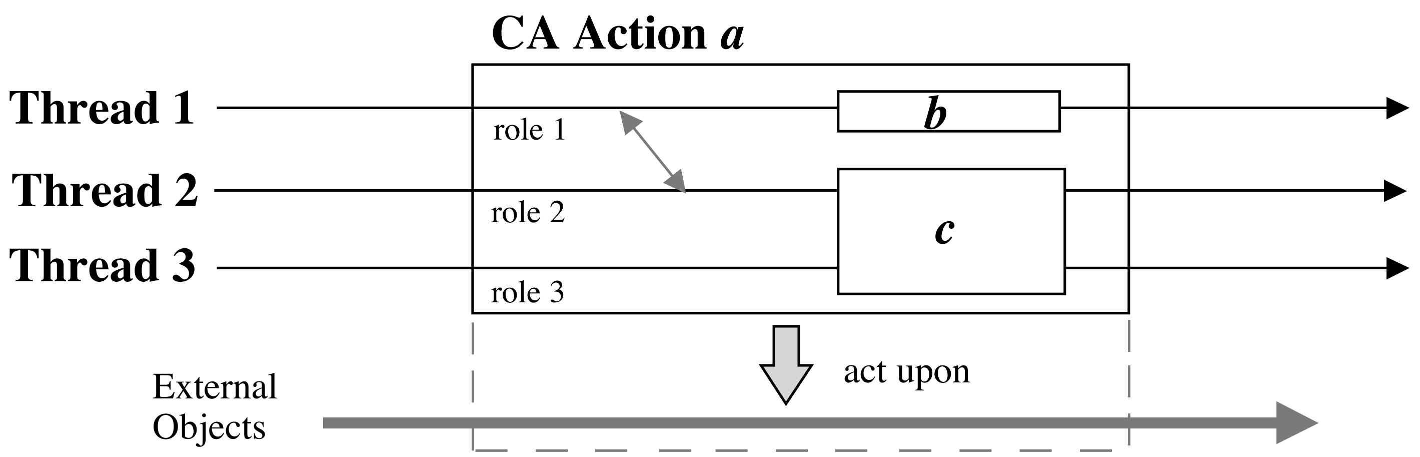


Within a CA Action, each of the participating threads play a defined "role".

**1) *Cooperation of Roles (i.e. Participating threads)*--
is via local shared objects**

**2) *Action upon External Objects* --
Operations on the external objects can be invoked by roles**

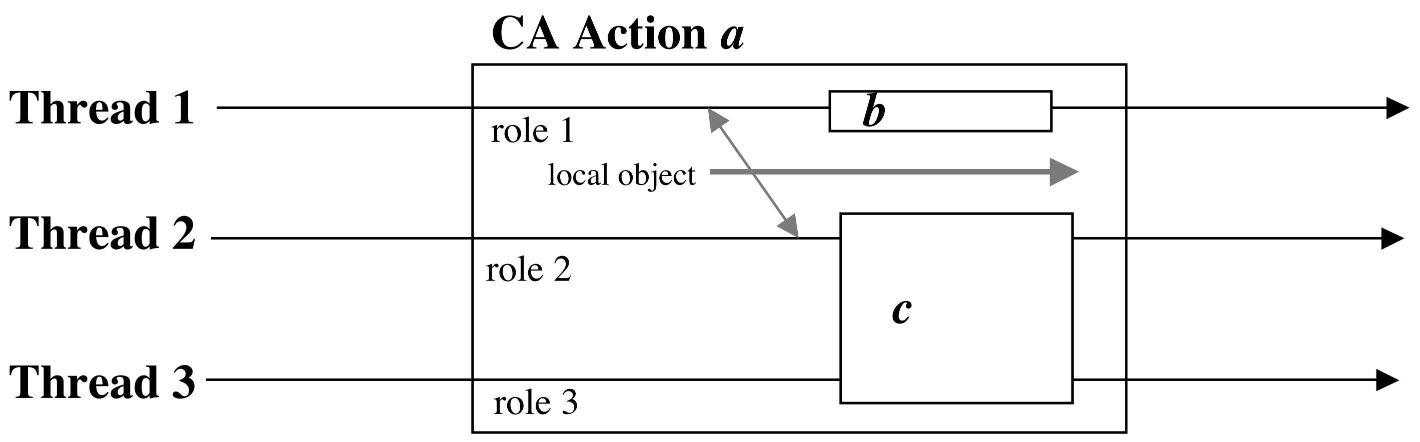
Nested CA Actions



Key properties:

1) Once action *a* obtains control over the external objects, the action can act upon them;
2) In line with normal rules for nested transactions, the external objects are still unaccessible for nested actions *b* and *c* until the nested actions obtain their own control over them;
3) The ACID properties of a nested action are always kept no matter how the action gains the external objects, i.e. competitively or cooperatively

Local Objects



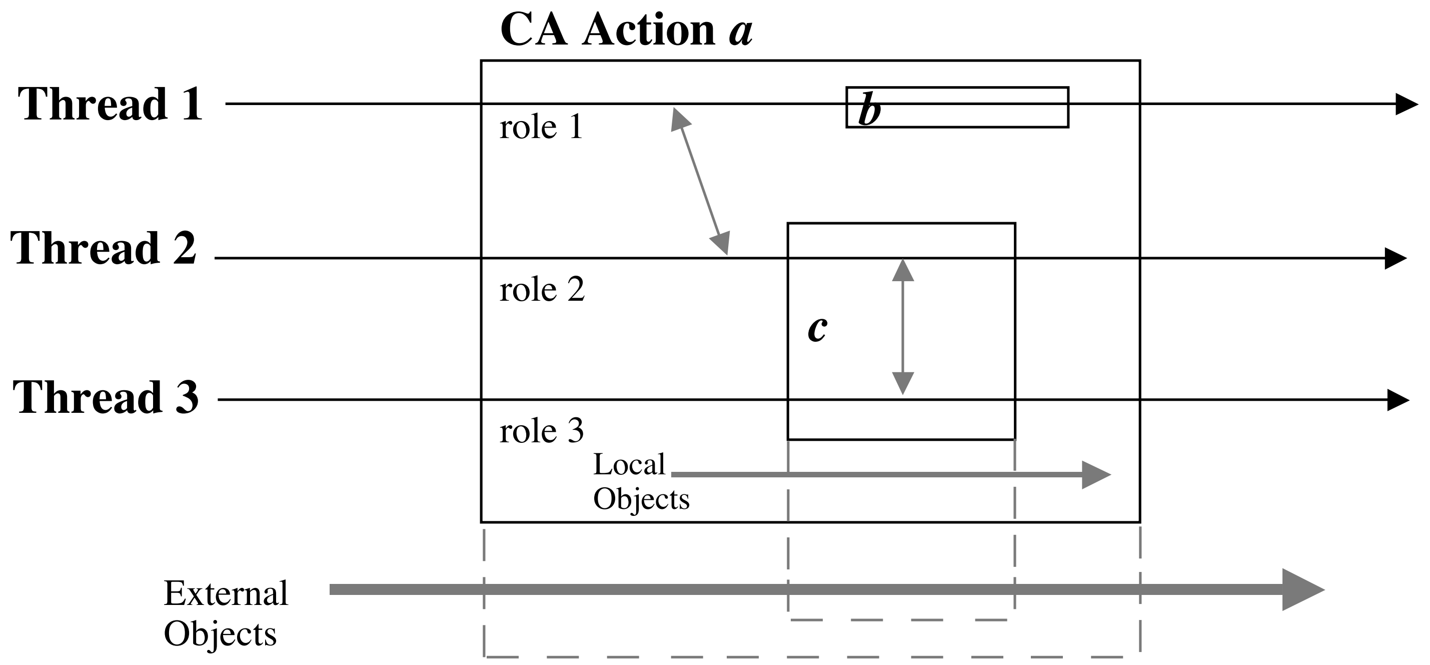
Key properties:

1) "Local" means the world outside a given action knows nothing about the existence of such objects; they are created and deleted within the action

2) However, such local objects are external to any nested actions. A nested action has to obtain (from the containing action) control over them before it can act upon them with the ACID properties

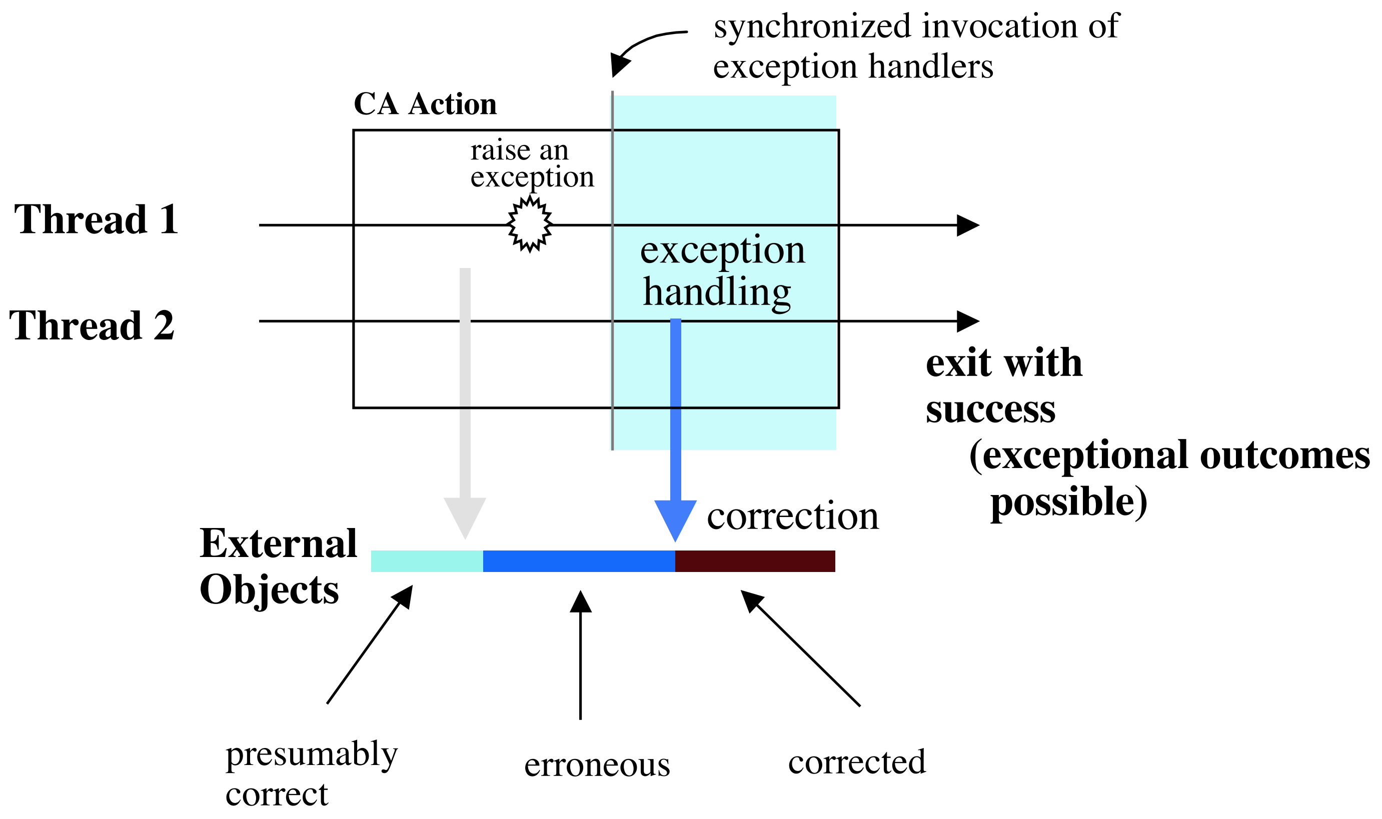
Recursive View

The CA action concept is thus recursive:

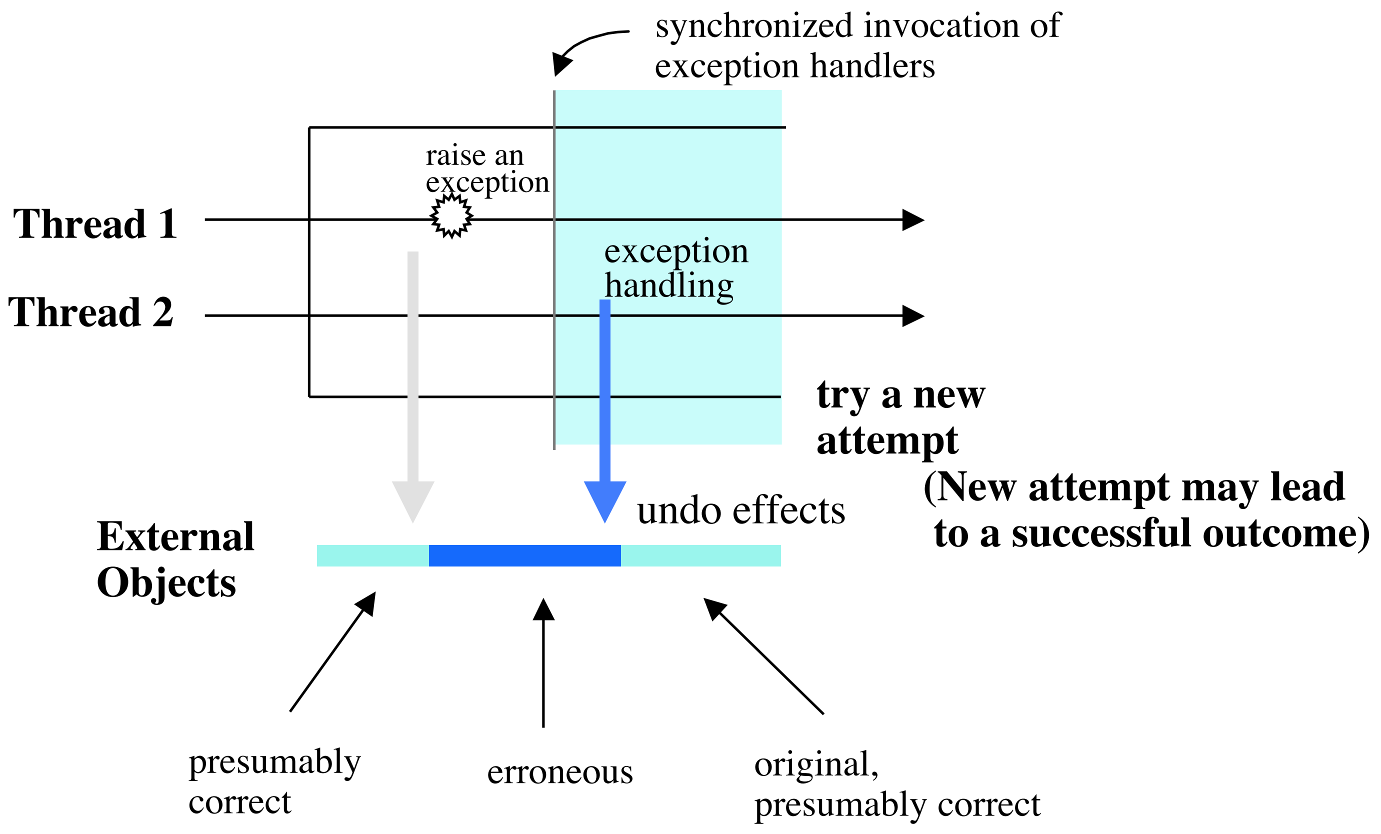


*** The external objects of the nested action *c* consist of both external objects and local objects of the containing action *a***

Forward Recovery of Internal Activities



Backward Recovery of Internal Activities



Fault Tolerance

1) Tolerating hardware-related faults can use standard transaction-based system techniques;

2) Recovery blocks or N-version programming techniques can be embedded with the CA action structure to tolerate software design faults:

For example, when a CA action fails with an *abort* exception, a new action that has the same functionality but with diverse design can try a further attempt in the hope that the failure will not occur

It is also possible to execute several variants of an action in parallel to achieve software fault masking and provide a timely outcome