Wikipedia
The Free Encyclopedia

Join-calculus

The **join-calculus** is a process calculus developed at INRIA. The join-calculus was developed to provide a formal basis for the design of distributed programming languages, and therefore intentionally avoids communications constructs found in other process calculi, such as rendezvous communications, which are difficult to implement in a distributed setting.[1] Despite this limitation, the join-calculus is as expressive as the full π-calculus. Encodings of the π-calculus in the join-calculus, and vice versa, have been demonstrated.[2]

The join-calculus is a member of the π-calculus family of process calculi, and can be considered, at its core, an asynchronous π-calculus with several strong restrictions:[3]

- Scope restriction, reception, and replicated reception are syntactically merged into a single construct, the *definition*;
- Communication occurs only on defined names;
- For every defined name there is exactly one replicated reception.

However, as a language for programming, the join-calculus offers at least one convenience over the π-calculus — namely the use of *multi-way join patterns*, the ability to match against messages from multiple channels simultaneously.[4]

Implementations

Languages based on the join-calculus

The join-calculus programming language is a new language based on the join-calculus process calculus. It is implemented as an interpreter written in OCaml, and supports statically typed distributed programming, transparent remote communication, agent-based mobility, and some failure-detection.[5]

- Though not explicitly based on join-calculus, the rule system of CLIPS implements it if every rule deletes its inputs when triggered (retracts the relevant facts when fired).

Many implementations of the join-calculus were made as extensions of existing programming languages:

- JoCaml is a version of OCaml extended with join-calculus primitives
- Polyphonic C# and its successor Cω extend C#
- MC# and Parallel C# extend Polyphonic C#
- Join Java extends Java
- A Concurrent Basic proposal that uses Join-calculus

- JErlang (the J is for Join, erjang is Erlang for the JVM)[6]

Embeddings in other programming languages

These implementations do not change the underlying programming language but introduce join calculus operations through a custom library or DSL:

- The ScalaJoins and the Chymyst (https://github.com/Chymyst/Chymyst) libraries are in Scala
- JoinHs (http://joinhs.sourceforge.net/) by Einar Karttunen and syallop/Join-Language (https://github.com/syallop/Join-Language) by Samuel Yallop are DSLs for Join calculus in Haskell
- Joinads - various implementations of join calculus in F#
- CocoaJoin is an experimental implementation in Objective-C for iOS and Mac OS X
- The Join Python library in Python 3[7]
- C++ via Boost[8] (for boost from 2009, ca. v. 40, current (Dec '19) is 72).

References

1. Cedric Fournet, Georges Gonthier (1995). "The reflexive CHAM and the join-calculus" (http://citeseer.ist.psu.edu/fournet95reflexive.html). {{cite journal}}: Cite journal requires |journal= (help), pg. 1
2. Cedric Fournet, Georges Gonthier (1995). "The reflexive CHAM and the join-calculus" (http://citeseer.ist.psu.edu/fournet95reflexive.html). {{cite journal}}: Cite journal requires |journal= (help), pg. 2
3. Cedric Fournet, Georges Gonthier (1995). "The reflexive CHAM and the join-calculus" (http://citeseer.ist.psu.edu/fournet95reflexive.html). {{cite journal}}: Cite journal requires |journal= (help), pg. 19
4. Petricek, Tomas. "TryJoinads (IV.) - Concurrency using join calculus" (http://tomasp.net/blog/joinads-join-calculus.aspx/). *tomasp.net*. Retrieved 2023-01-24.
5. Cedric Fournet, Georges Gonthier (2000). "The Join Calculus: A Language for Distributed Mobile Programming" (https://www.microsoft.com/en-us/research/publication/join-calculus-language-distributed-mobile-programming/): 268–332. {{cite journal}}: Cite journal requires |journal= (help)
6. "JErlang: Erlang with Joins" (https://web.archive.org/web/20171208175247/http://www.doc.ic.ac.uk/~susan/jerlang/). Archived from the original (http://www.doc.ic.ac.uk/~susan/jerlang/) on 2017-12-08. Retrieved 2015-04-18.
7. Join Python, Join-calculus for Python by Mattias Andree (https://github.com/maandree/join-python/blob/master/join-python.pdf)
8. Yigong Liu - Join-Asynchronous Message Coordination and Concurrency Library (http://channel.sourceforge.net/boost_join/libs/join/doc/boost_join_design.html)

External links

- INRIA, Join Calculus homepage (http://moscova.inria.fr/join/index.shtml)
- Microsoft Research, The Join Calculus: a Language for Distributed Mobile Programming (https://www.microsoft.com/en-us/research/wp-content/uploads/2017/01/join-tutorial.pdf)