CS 220: Survey of Programming Languages

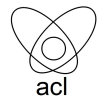
Lecture Slides

Concurrent Programming Languages

Jan Michael C. Yap

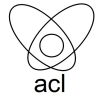
Algorithms and Complexity Laboratory
Department of Computer Science
University of the Philippines, Diliman
jcyap@dcs.upd.edu.ph

Session 8

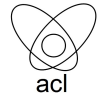


Concurrent Programming Languages

Concurrent Programming Languages

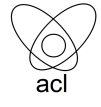


Concurrent programming



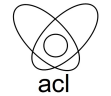
Concurrent programming

- Concurrent programmming focuses on allowing multiple operations to occur simultaneously, possibly in cooperation with each other

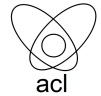


Concurrent programming

- Concurrent programmming focuses on allowing multiple operations to occur simultaneously, possibly in cooperation with each other
- This can be done either with computations sharing common memory pool (parallel) or with each computation having their own memory pool (distributed)

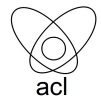


Characteristics of concurrent PL



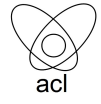
Characteristics of concurrent PL

- Language/syntax structure heavily borrows from imperative, functional, and /or OO paradigms
  - And oftentimes, built on top of those paradigms



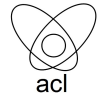
Characteristics of concurrent PL

- Language/syntax structure heavily borrows from imperative, functional, and /or OO paradigms
  - And oftentimes, built on top of those paradigms
- Computation is represented internally as a thread, which may interact with other threads



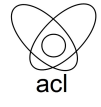
Characteristics of concurrent PL

- Language/syntax structure heavily borrows from imperative, functional, and /or OO paradigms
  - And oftentimes, built on top of those paradigms
- Computation is represented internally as a thread, which may interact with other threads
- One particular concern in concurrent PLs is facilitating access to shared resources to several (potentially competing) threads



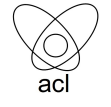
Characteristics of concurrent PL

- Language/syntax structure heavily borrows from imperative, functional, and /or OO paradigms
  - And oftentimes, built on top of those paradigms
- Computation is represented internally as a thread, which may interact with other threads
- One particular concern in concurrent PLs is facilitating access to shared resources to several (potentially competing) threads
  - One way to do this is via definition of critical regions/sections, and have synchronization of thread access to such facilitated by semaphores

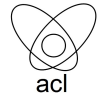


Characteristics of concurrent PL

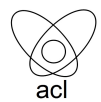
- Language/syntax structure heavily borrows from imperative, functional, and /or OO paradigms
  - And oftentimes, built on top of those paradigms
- Computation is represented internally as a thread, which may interact with other threads
- One particular concern in concurrent PLs is facilitating access to shared resources to several (potentially competing) threads
  - One way to do this is via definition of critical regions/sections, and have synchronization of thread access to such facilitated by semaphores
  - An alternative to using semaphores and critical sections us via the use of monitors



Concurrent Programming Languages



Background

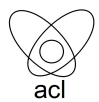


---

http://www.erlang.org/course/history.html

Background

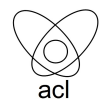
- Stands for Ericsson Language



http://www.erlang.org/course/history.html

Background

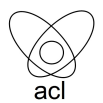
- Stands for Ericsson Language
- Developed in 1987 as an alternative programming language that can be "... a very high level symbolic language in order to achive (sic) productivity gains"



http://www.erlang.org/course/history.html

Background

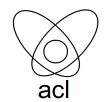
- Stands for Ericsson Language
- Developed in 1987 as an alternative programming language that can be "... a very high level symbolic language in order to achive (sic) productivity gains"
  - "Experiments with programming of telecom using $>$ 20 different languages" were made, including LISP



http://www.erlang.org/course/history.html

Background

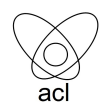
- Stands for Ericsson Language
- Developed in 1987 as an alternative programming language that can be "... a very high level symbolic language in order to achive (sic) productivity gains"
  - "Experiments with programming of telecom using $>$ 20 different languages" were made, including LISP
  - Symbolic language implied that Erlang can also be categorized as a functional PL



http://www.erlang.org/course/history.html

Background

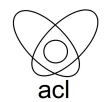
- Stands for Ericsson Language
- Developed in 1987 as an alternative programming language that can be "... a very high level symbolic language in order to achive (sic) productivity gains"
  - "Experiments with programming of telecom using $>$ 20 different languages" were made, including LISP
  - Symbolic language implied that Erlang can also be categorized as a functional PL
- Design principles



http://www.erlang.org/course/history.html

Background

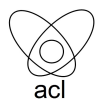
- Stands for Ericsson Language
- Developed in 1987 as an alternative programming language that can be "... a very high level symbolic language in order to achive (sic) productivity gains"
  - "Experiments with programming of telecom using $>$ 20 different languages" were made, including LISP
  - Symbolic language implied that Erlang can also be categorized as a functional PL
- Design principles
  - "The language must contain primitives for concurrency and error recovery, and the execution model must not have back-tracking."



http://www.erlang.org/course/history.html

Background

- Stands for Ericsson Language
- Developed in 1987 as an alternative programming language that can be "... a very high level symbolic language in order to achive (sic) productivity gains"
  - "Experiments with programming of telecom using $>$ 20 different languages" were made, including LISP
  - Symbolic language implied that Erlang can also be categorized as a functional PL
- Design principles
  - "The language must contain primitives for concurrency and error recovery, and the execution model must not have back-tracking."
  - "It must also have a granularity of concurrency such that one asyncronous telephony process is represented by one process in the language."



http://www.erlang.org/course/history.html

Evaluation

- Data types[1,2]

[1] https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl
[2] http://www.erlang.org/doc/reference_manual/data_types.html
[3] Manually counted

Evaluation

- Data types[1,2]
  - Integer, single and double precision floating point real numbers, logic, character and string primitive data types

[1] https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl
[2] http://www.erlang.org/doc/reference_manual/data_types.html
[3] Manually counted

Evaluation

- Data types[1,2]
  - Integer, single and double precision floating point real numbers, logic, character and string primitive data types
  - Lists, tuples (although to be deprecated soon), records, maps, and process as derived types

[1] https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl
[2] http://www.erlang.org/doc/reference_manual/data_types.html
[3] Manually counted

Evaluation

- Data types[1,2]
  - Integer, single and double precision floating point real numbers, logic, character and string primitive data types
  - Lists, tuples (although to be deprecated soon), records, maps, and process as derived types
  - Functions are treated as objects

[1] https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl
[2] http://www.erlang.org/doc/reference_manual/data_types.html
[3] Manually counted

Evaluation

- Data types[1,2]
  - Integer, single and double precision floating point real numbers, logic, character and string primitive data types
  - Lists, tuples (although to be deprecated soon), records, maps, and process as derived types
  - Functions are treated as objects
- Syntax design[1,3]:

[1] https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl
[2] http://www.erlang.org/doc/reference_manual/data_types.html
[3] Manually counted

Evaluation

- Data types[1,2]
  - Integer, single and double precision floating point real numbers, logic, character and string primitive data types
  - Lists, tuples (although to be deprecated soon), records, maps, and process as derived types
  - Functions are treated as objects
- Syntax design[1,3]:
  - Production rules: 255 (excluding deprecated features, also BNF is normalized)
  - Keywords: 34

[1] https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl
[2] http://www.erlang.org/doc/reference_manual/data_types.html
[3] Manually counted



Evaluation

- Data types[1,2]
  - Integer, single and double precision floating point real numbers, logic, character and string primitive data types
  - Lists, tuples (although to be deprecated soon), records, maps, and process as derived types
  - Functions are treated as objects
- Syntax design[1,3]:
  - Production rules: 255 (excluding deprecated features, also BNF is normalized)
  - Keywords: 34
  - Erlang adheres to functional programming paradigm concepts, and thus is a very orthogonal language, and the structure of statements and expressions have (little to) no special cases as per the grammar
    - Reminiscent of Haskell language (though probably unintentional)

[1] https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl
[2] http://www.erlang.org/doc/reference_manual/data_types.html
[3] Manually counted

Evaluation

- Abstraction:

[4] http://www.erlang.org/doc/reference_manual/modules.html
[5] http://www.erlang.org/doc/reference_manual/data_types.html
[6] http://rosettacode.org/wiki/Scope_modifiers#Erlang
[7] http://erlang.org/doc/man/erlang.html

Evaluation

- Abstraction:
  - Modules and records can be used to create data structures[4,5]

[4] http://www.erlang.org/doc/reference_manual/modules.html
[5] http://www.erlang.org/doc/reference_manual/data_types.html
[6] http://rosettacode.org/wiki/Scope_modifiers#Erlang
[7] http://erlang.org/doc/man/erlang.html

Evaluation

- Abstraction:
  - Modules and records can be used to create data structures[4,5]
  - "Erlang is lexically scoped. Variables, which must begin with an upper case letter, are only available inside their functions. Functions are only available inside their modules [, u]nless they are exported."[6]

[4] http://www.erlang.org/doc/reference_manual/modules.html
[5] http://www.erlang.org/doc/reference_manual/data_types.html
[6] http://rosettacode.org/wiki/Scope_modifiers#Erlang
[7] http://erlang.org/doc/man/erlang.html

Evaluation

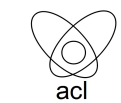
- Abstraction:
  - Modules and records can be used to create data structures[4,5]
  - "Erlang is lexically scoped. Variables, which must begin with an upper case letter, are only available inside their functions. Functions are only available inside their modules [, u]nless they are exported."[6]
- Expressivity: Has a rich set of functions in the core module, most pertinent are optimization functions for handling data structures and concurrency [7]

[4] http://www.erlang.org/doc/reference_manual/modules.html
[5] http://www.erlang.org/doc/reference_manual/data_types.html
[6] http://rosettacode.org/wiki/Scope_modifiers#Erlang
[7] http://erlang.org/doc/man/erlang.html

Evaluation

- Exception handling:[8]

[8] http://erlang.org/doc/reference_manual/errors.html
[9] http://erlang.org/pipermail/erlang-questions/2013-March/072760.html



Evaluation

- Exception handling:[8]
  - Has a `try` and `catch` statements to handle exceptions in code

[8] http://erlang.org/doc/reference_manual/errors.html
[9] http://erlang.org/pipermail/erlang-questions/2013-March/072760.html

Evaluation

- Exception handling:[8]
  - Has a `try` and `catch` statements to handle exceptions in code
  - Termination of processes can be made by signalling an exit reason

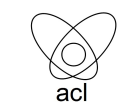
[8] http://erlang.org/doc/reference_manual/errors.html
[9] http://erlang.org/pipermail/erlang-questions/2013-March/072760.html

Evaluation

- Exception handling:[8]
  - Has a `try` and `catch` statements to handle exceptions in code
  - Termination of processes can be made by signalling an exit reason
- Restricted aliasing: "Erlang uses pass-by-value."[9]

[8] http://erlang.org/doc/reference_manual/errors.html
[9] http://erlang.org/pipermail/erlang-questions/2013-March/072760.html





Evaluation

- Efficiency [10,11,12]

[10] http://www.erlang.org/doc/efficiency_guide/myths.html
[11] http://www.erlang.org/doc/efficiency_guide/commoncaveats.html
[12] http://learnyousomeerlang.com/types-or-lack-thereof

Evaluation

- Efficiency [10,11,12]
  - Employs dynamic type-checking

[10] http://www.erlang.org/doc/efficiency_guide/myths.html
[11] http://www.erlang.org/doc/efficiency_guide/commoncaveats.html
[12] http://learnyousomeerlang.com/types-or-lack-thereof

Evaluation

- Efficiency [10,11,12]
  - Employs dynamic type-checking
  - Erlang is an interpreted language

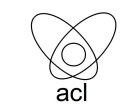
[10] http://www.erlang.org/doc/efficiency_guide/myths.html
[11] http://www.erlang.org/doc/efficiency_guide/commoncaveats.html
[12] http://learnyousomeerlang.com/types-or-lack-thereof



Evaluation

- Efficiency [10,11,12]
  - Employs dynamic type-checking
  - Erlang is an interpreted language
  - Anonymous functions (`funs`) are notoriously slow, although improved in subsequent versions

[10] http://www.erlang.org/doc/efficiency_guide/myths.html
[11] http://www.erlang.org/doc/efficiency_guide/commoncaveats.html
[12] http://learnyousomeerlang.com/types-or-lack-thereof

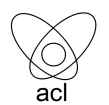


Evaluation

- Efficiency [10,11,12]
  - Employs dynamic type-checking
  - Erlang is an interpreted language
  - Anonymous functions (`funs`) are notoriously slow, although improved in subsequent versions
  - Tail-recursive functions are inherently faster than recursive ones.

[10] http://www.erlang.org/doc/efficiency_guide/myths.html
[11] http://www.erlang.org/doc/efficiency_guide/commoncaveats.html
[12] http://learnyousomeerlang.com/types-or-lack-thereof

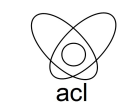




Evaluation

- Efficiency [10,11,12]
  - Employs dynamic type-checking
  - Erlang is an interpreted language
  - Anonymous functions (`funs`) are notoriously slow, although improved in subsequent versions
  - Tail-recursive functions are inherently faster than recursive ones.
  - "[S]tring handling could be slow if done improperly."

[10] http://www.erlang.org/doc/efficiency_guide/myths.html
[11] http://www.erlang.org/doc/efficiency_guide/commoncaveats.html
[12] http://learnyousomeerlang.com/types-or-lack-thereof



Evaluation

- Efficiency [10,11,12]
  - Employs dynamic type-checking
  - Erlang is an interpreted language
  - Anonymous functions (`funs`) are notoriously slow, although improved in subsequent versions
  - Tail-recursive functions are inherently faster than recursive ones.
  - "[S]tring handling could be slow if done improperly.'
  - "The `timer` module uses a separate process to manage the timers, and that process can easily become overloaded if many processes create and cancel timers frequently..."

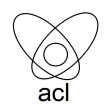
[10] http://www.erlang.org/doc/efficiency_guide/myths.html
[11] http://www.erlang.org/doc/efficiency_guide/commoncaveats.html
[12] http://learnyousomeerlang.com/types-or-lack-thereof

Evaluation

- Efficiency [10,11,12]
  - Employs dynamic type-checking
  - Erlang is an interpreted language
  - Anonymous functions (`funs`) are notoriously slow, although improved in subsequent versions
  - Tail-recursive functions are inherently faster than recursive ones.
  - "[S]tring handling could be slow if done improperly.'
  - "The `timer` module uses a separate process to manage the timers, and that process can easily become overloaded if many processes create and cancel timers frequently..."
  - "Atoms are not garbage-collected. Once an atom is created, it will never be removed."

[10] http://www.erlang.org/doc/efficiency_guide/myths.html
[11] http://www.erlang.org/doc/efficiency_guide/commoncaveats.html
[12] http://learnyousomeerlang.com/types-or-lack-thereof



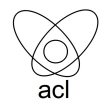


Evaluation

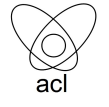
- Efficiency [10,11,12]
  - Employs dynamic type-checking
  - Erlang is an interpreted language
  - Anonymous functions (`funs`) are notoriously slow, although improved in subsequent versions
  - Tail-recursive functions are inherently faster than recursive ones.
  - "[S]tring handling could be slow if done improperly.'
  - "The `timer` module uses a separate process to manage the timers, and that process can easily become overloaded if many processes create and cancel timers frequently..."
  - "Atoms are not garbage-collected. Once an atom is created, it will never be removed."
  - "The time for calculating the length of a list is proportional to the length of the list, as opposed to [getting size of tuples, bytes, and bits] which all execute in constant time."

[10] http://www.erlang.org/doc/efficiency_guide/myths.html
[11] http://www.erlang.org/doc/efficiency_guide/commoncaveats.html
[12] http://learnyousomeerlang.com/types-or-lack-thereof





Concurrent Programming Languages



Background

[13] http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo
[14] http://www.vitanuova.com/inferno/index.html
[15] http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Background

- "Limbo is a programming language intended for applications running distributed systems on small computers."[13]

[13] http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo
[14] http://www.vitanuova.com/inferno/index.html
[15] http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Background

- "Limbo is a programming language intended for applications running distributed systems on small computers."[13]
- "Life is made easier for the programmer with features such as automatic garbage collection, compile and runtime type checking and simple creation of multiple processes (threads) and communication between them"[14]

[13] http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo
[14] http://www.vitanuova.com/inferno/index.html
[15] http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Background

- "Limbo is a programming language intended for applications running distributed systems on small computers."[13]
- "Life is made easier for the programmer with features such as automatic garbage collection, compile and runtime type checking and simple creation of multiple processes (threads) and communication between them"[14]
- Used to develop the Inferno operating system, which is meant for building cross-platform distributed systems[14]

[13] http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo
[14] http://www.vitanuova.com/inferno/index.html
[15] http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Background

- "Limbo is a programming language intended for applications running distributed systems on small computers."[13]
- "Life is made easier for the programmer with features such as automatic garbage collection, compile and runtime type checking and simple creation of multiple processes (threads) and communication between them"[14]
- Used to develop the Inferno operating system, which is meant for building cross-platform distributed systems[14]
- Built on top of C, and adopts a couple of C functionalities[15]

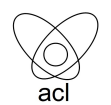
[13] http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo
[14] http://www.vitanuova.com/inferno/index.html
[15] http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

- Data types[16]

[16] http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo
[17] Manually counted
[18] https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default
[19] http://www.vitanuova.com/inferno/limbo.html





Evaluation

- Data types[16]
  - Integers, 64-bit long floating point real numbers, and strings as primitive types

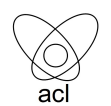
[16] http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo
[17] Manually counted
[18] https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default
[19] http://www.vitanuova.com/inferno/limbo.html

Evaluation

- Data types[16]
  - Integers, 64-bit long floating point real numbers, and strings as primitive types
  - Arrays, tuples, and lists as derived types

[16] http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo
[17] Manually counted
[18] https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default
[19] http://www.vitanuova.com/inferno/limbo.html



Evaluation

- Data types[16]
  - Integers, 64-bit long floating point real numbers, and strings as primitive types
  - Arrays, tuples, and lists as derived types
  - A channel type pertains to " a communication mechanism capable of sending and receiving objects of the specified type to another agent in the system"

[16] http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo
[17] Manually counted
[18] https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default
[19] http://www.vitanuova.com/inferno/limbo.html

Evaluation

- Data types[16]
  - Integers, 64-bit long floating point real numbers, and strings as primitive types
  - Arrays, tuples, and lists as derived types
  - A channel type pertains to " a communication mechanism capable of sending and receiving objects of the specified type to another agent in the system"
- Syntax design:

[16] http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo
[17] Manually counted
[18] https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default
[19] http://www.vitanuova.com/inferno/limbo.html

Evaluation

- Data types[16]
  - Integers, 64-bit long floating point real numbers, and strings as primitive types
  - Arrays, tuples, and lists as derived types
  - A channel type pertains to " a communication mechanism capable of sending and receiving objects of the specified type to another agent in the system"
- Syntax design:
  - Production rules:72[16,17]
  - Keywords: 46[17,18]

[16] http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo
[17] Manually counted
[18] https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default
[19] http://www.vitanuova.com/inferno/limbo.html

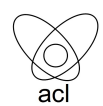
Evaluation

- Data types[16]
  - Integers, 64-bit long floating point real numbers, and strings as primitive types
  - Arrays, tuples, and lists as derived types
  - A channel type pertains to " a communication mechanism capable of sending and receiving objects of the specified type to another agent in the system"
- Syntax design:
  - Production rules:72[16,17]
  - Keywords: 46[17,18]
  - "Syntactically similar to C..."[19]

[16] http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo
[17] Manually counted
[18] https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default
[19] http://www.vitanuova.com/inferno/limbo.html

Evaluation

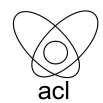
- Abstraction: Functions and other resources one wishes to be made public must be listed inside a `module` declaration



http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

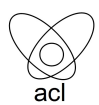
- Abstraction: Functions and other resources one wishes to be made public must be listed inside a `module` declaration
- Expressivity: Has extensive standard libraries, notable of which is for handling cross-platform development, GUI programming, and regular expressions



http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

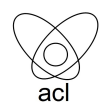
- Abstraction: Functions and other resources one wishes to be made public must be listed inside a `module` declaration
- Expressivity: Has extensive standard libraries, notable of which is for handling cross-platform development, GUI programming, and regular expressions
- Exception handling



http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

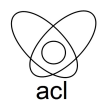
- Abstraction: Functions and other resources one wishes to be made public must be listed inside a `module` declaration
- Expressivity: Has extensive standard libraries, notable of which is for handling cross-platform development, GUI programming, and regular expressions
- Exception handling
  - Exception handling can be made via the `exception` expression



http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

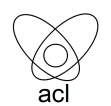
- Abstraction: Functions and other resources one wishes to be made public must be listed inside a `module` declaration
- Expressivity: Has extensive standard libraries, notable of which is for handling cross-platform development, GUI programming, and regular expressions
- Exception handling
  - Exception handling can be made via the `exception` expression
  - Capacity to create custom exception



http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

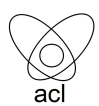
Evaluation

- Abstraction: Functions and other resources one wishes to be made public must be listed inside a `module` declaration
- Expressivity: Has extensive standard libraries, notable of which is for handling cross-platform development, GUI programming, and regular expressions
- Exception handling
  - Exception handling can be made via the `exception` expression
  - Capacity to create custom exception
  - Exceptions can be signalled via a `raise` statement



http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

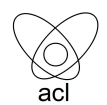


---

http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

- Restricted aliasing:

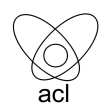


---

http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

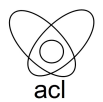
- Restricted aliasing:
  - Employs pass-by-value and pass-by-reference



http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

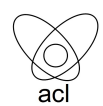
- Restricted aliasing:
  - Employs pass-by-value and pass-by-reference
  - "Pointers... are restricted compared to C: they can only refer to adt values on the heap."



http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

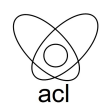
- Restricted aliasing:
  - Employs pass-by-value and pass-by-reference
  - "Pointers... are restricted compared to C: they can only refer to adt values on the heap."
  - "There is no & (address of) operator, nor is address arithmetic possible."



http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

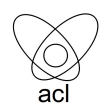
- Restricted aliasing:
  - Employs pass-by-value and pass-by-reference
  - "Pointers... are restricted compared to C: they can only refer to adt values on the heap."
  - "There is no & (address of) operator, nor is address arithmetic possible."
  - "Arrays are also reference types, however, and since array slicing is supported, that replaces many of C's pointer constructions."



http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

- Efficiency:

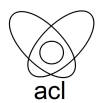


---

http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

- Efficiency:
  - Employs static and dynamic type checking.

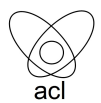


---

http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

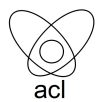
- Efficiency:
  - Employs static and dynamic type checking.
  - "In its implementation for the Inferno operating system, object programs generated by the Limbo compiler run using an interpreter for a fixed virtual machine. "



http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Evaluation

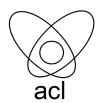
- Efficiency:
  - Employs static and dynamic type checking.
  - "In its implementation for the Inferno operating system, object programs generated by the Limbo compiler run using an interpreter for a fixed virtual machine. "
    - "Inferno and its accompanying virtual machine run either stand-alone on bare hardware or as an application under conventional operating systems"



http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

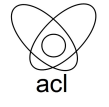
Evaluation

- Efficiency:
  - Employs static and dynamic type checking.
  - "In its implementation for the Inferno operating system, object programs generated by the Limbo compiler run using an interpreter for a fixed virtual machine. "
    - "Inferno and its accompanying virtual machine run either stand-alone on bare hardware or as an application under conventional operating systems"
  - "For most architectures, including Intel x86, ARM, PowerPC, MIPS and Sparc, Limbo object programs are transformed on-the-fly into instructions for the underlying hardware."

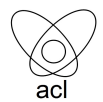


http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent

Concurrent Programming Languages



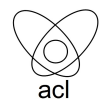
Background



http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf

Background

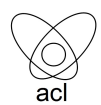
- Orc is a concurrent and distributed programming language "which provides uniform access to computational services, including distributed communication and data manipulation, through sites"



http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf

Background

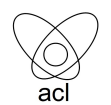
- Orc is a concurrent and distributed programming language "which provides uniform access to computational services, including distributed communication and data manipulation, through sites"
- Orchestration, i.e. the integration and synchronization of sites, is performed such that computations with "delays associated with communication, unreliability of servers, and competition for resources from multiple clients" efficiently performed



http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf

Background

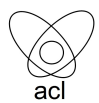
- Orc is a concurrent and distributed programming language "which provides uniform access to computational services, including distributed communication and data manipulation, through sites"
- Orchestration, i.e. the integration and synchronization of sites, is performed such that computations with "delays associated with communication, unreliability of servers, and competition for resources from multiple clients" efficiently performed
- It can be used as a general purpose, web scripting, and executable specification language.



http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf

Background

- Orc is a concurrent and distributed programming language "which provides uniform access to computational services, including distributed communication and data manipulation, through sites"
- Orchestration, i.e. the integration and synchronization of sites, is performed such that computations with "delays associated with communication, unreliability of servers, and competition for resources from multiple clients" efficiently performed
- It can be used as a general purpose, web scripting, and executable specification language.
- Borrows some elements from, and is built on top of, Java



http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf

Evaluation

- Data types[21]

[21] https://orc.csres.utexas.edu/documentation/html/userguide/userguide.html
[22] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.syntax.EBNF.html
[23] Manually counted
[24] http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf

Evaluation

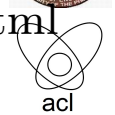
- Data types[21]
  - Integer, string, logical, and signals as primitive data types

[21] https://orc.csres.utexas.edu/documentation/html/userguide/userguide.html
[22] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.syntax.EBNF.html
[23] Manually counted
[24] http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf

Evaluation

- Data types[21]
  - Integer, string, logical, and signals as primitive data types
  - Tuples, lists, and patterns as derived types

[21] https://orc.csres.utexas.edu/documentation/html/userguide/userguide.html
[22] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.syntax.EBNF.html
[23] Manually counted
[24] http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf

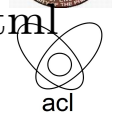


Evaluation

- Data types[21]
  - Integer, string, logical, and signals as primitive data types
  - Tuples, lists, and patterns as derived types
- Syntax design:

[21] https://orc.csres.utexas.edu/documentation/html/userguide/userguide.html
[22] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.syntax.EBNF.html
[23] Manually counted
[24] http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf

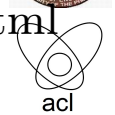




Evaluation

- Data types[21]
  - Integer, string, logical, and signals as primitive data types
  - Tuples, lists, and patterns as derived types
- Syntax design:
  - Production rules: 69[22,23]
  - Keywords: 20

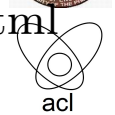
[21] https://orc.csres.utexas.edu/documentation/html/userguide/userguide.html
[22] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.syntax.EBNF.html
[23] Manually counted
[24] http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf



Evaluation

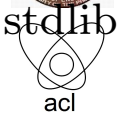
- Data types[21]
  - Integer, string, logical, and signals as primitive data types
  - Tuples, lists, and patterns as derived types
- Syntax design:
  - Production rules: 69[22,23]
  - Keywords: 20
  - Design is very simple, with emphasis on facilitating concurrent computations [24]

[21] https://orc.csres.utexas.edu/documentation/html/userguide/userguide.html
[22] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.syntax.EBNF.html
[23] Manually counted
[24] http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf



Evaluation

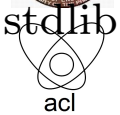
- Abstraction:



[25] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.declarations.defclass.h
[26] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.sites.custom.html#ref.
[27] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.stdlib.html#ref.stdlib
[28] https://orc.csres.utexas.edu/papers/OrcExceptionSemantics.pdf
[29] https://orc.csres.utexas.edu/documentation/html/refmanual/index.html

Evaluation

- Abstraction:
  - Definition of classes, ala-OOPL supported[25]



[25] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.declarations.defclass.h
[26] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.sites.custom.html#ref.
[27] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.stdlib.html#ref.stdlib
[28] https://orc.csres.utexas.edu/papers/OrcExceptionSemantics.pdf
[29] https://orc.csres.utexas.edu/documentation/html/refmanual/index.html

Evaluation

- Abstraction:
  - Definition of classes, ala-OOPL supported[25]
  - Definition of sites, which are the primary interface of a program to its environment, is done similarly as definition of classes[26]

[25] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.declarations.defclass.h
[26] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.sites.custom.html#ref.
[27] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.stdlib.html#ref.stdlib
[28] https://orc.csres.utexas.edu/papers/OrcExceptionSemantics.pdf
[29] https://orc.csres.utexas.edu/documentation/html/refmanual/index.html

Evaluation

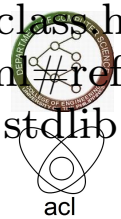
- Abstraction:
  - Definition of classes, ala-OOPL supported[25]
  - Definition of sites, which are the primary interface of a program to its environment, is done similarly as definition of classes[26]
- Expressivity: Standard library consists mostly of facilitating data structure construction, and resource synchronization utilities, as well as Web/network programming utilities[27]

[25] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.declarations.defclass.h
[26] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.sites.custom.html#ref.
[27] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.stdlib.html#ref.stdlib
[28] https://orc.csres.utexas.edu/papers/OrcExceptionSemantics.pdf
[29] https://orc.csres.utexas.edu/documentation/html/refmanual/index.html

Evaluation

- Abstraction:
  - Definition of classes, ala-OOPL supported[25]
  - Definition of sites, which are the primary interface of a program to its environment, is done similarly as definition of classes[26]
- Expressivity: Standard library consists mostly of facilitating data structure construction, and resource synchronization utilities, as well as Web/network programming utilities[27]
- Exception handling: Operational semantics for exception handling was proposed for Orc[28], but seems unimplemented as of current version (see Reference Manual[29]).

[25] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.declarations.defclass.h
[26] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.sites.custom.html#ref.
[27] http://orc.csres.utexas.edu/documentation/html/refmanual/ref.stdlib.html#ref.stdlib
[28] https://orc.csres.utexas.edu/papers/OrcExceptionSemantics.pdf
[29] https://orc.csres.utexas.edu/documentation/html/refmanual/index.html



Evaluation

- Restricted aliasing: Employs pass-by-value (in the same manner as Java)[30]

[30] https://orc.csres.utexas.edu/papers/OrcJSSM.pdf
[31] https://orc.csres.utexas.edu/download.shtml

Evaluation

- Restricted aliasing: Employs pass-by-value (in the same manner as Java)[30]
- Efficiency[31]

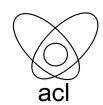
[30] https://orc.csres.utexas.edu/papers/OrcJSSM.pdf
[31] https://orc.csres.utexas.edu/download.shtml

Evaluation

- Restricted aliasing: Employs pass-by-value (in the same manner as Java)[30]
- Efficiency[31]
  - Employs dynamic type-checking by default, but can enforce static type-checking

[30] https://orc.csres.utexas.edu/papers/OrcJSSM.pdf
[31] https://orc.csres.utexas.edu/download.shtml



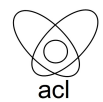


Evaluation

- Restricted aliasing: Employs pass-by-value (in the same manner as Java)[30]
- Efficiency[31]
  - Employs dynamic type-checking by default, but can enforce static type-checking
  - Orc is an interpreted language

[30] https://orc.csres.utexas.edu/papers/OrcJSSM.pdf
[31] https://orc.csres.utexas.edu/download.shtml



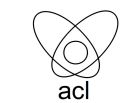


Evaluation

- Restricted aliasing: Employs pass-by-value (in the same manner as Java)[30]
- Efficiency[31]
  - Employs dynamic type-checking by default, but can enforce static type-checking
  - Orc is an interpreted language
  - No significant delays in the executions, and concurrency runs properly

[30] https://orc.csres.utexas.edu/papers/OrcJSSM.pdf
[31] https://orc.csres.utexas.edu/download.shtml



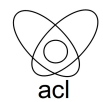



Evaluation

- Restricted aliasing: Employs pass-by-value (in the same manner as Java)[30]
- Efficiency[31]
  - Employs dynamic type-checking by default, but can enforce static type-checking
  - Orc is an interpreted language
  - No significant delays in the executions, and concurrency runs properly
  - Interpreter is developed using Java

[30] https://orc.csres.utexas.edu/papers/OrcJSSM.pdf
[31] https://orc.csres.utexas.edu/download.shtml





END OF SESSION 8