

# 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

Erlang

Limbo

Orc



# Concurrent Programming Languages

## Concurrent Programming Languages

Erlang

Limbo

Orc



# Concurrent programming



## Concurrent programming

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



## Concurrent programming

- **Concurrent** programming 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 is via the use of monitors



# Concurrent Programming Languages

Concurrent Programming Languages

Erlang

Limbo

Orc



# Background



## Background

- Stands for **E**ricsson **L**anguage



## Background

- Stands for **E**ricsson **L**anguage
- Developed in 1987 as an alternative programming language that can be “... a **very high level symbolic language in order to achieve (sic) productivity gains**”





## 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 achieve (sic) productivity gains**”
  - “Experiments with programming of telecom using **> 20 different languages**” were made, including LISP



## 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 achieve (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**



## 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 achieve (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



## 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 achieve (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.**”



## 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 achieve (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 asynchronous telephony process is represented by one process in the language.**”



## Evaluation

- Data types<sup>1,2</sup>

---

<sup>1</sup>[https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl\\_parse.yrl](https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl)

<sup>2</sup>[http://www.erlang.org/doc/reference\\_manual/data\\_types.html](http://www.erlang.org/doc/reference_manual/data_types.html)

<sup>3</sup>Manually counted



## Evaluation

- Data types<sup>1,2</sup>
  - Integer, single and double precision floating point real numbers, logic, character and string primitive data types

---

<sup>1</sup>[https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl\\_parse.yrl](https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl)

<sup>2</sup>[http://www.erlang.org/doc/reference\\_manual/data\\_types.html](http://www.erlang.org/doc/reference_manual/data_types.html)

<sup>3</sup>Manually counted



## Evaluation

- Data types<sup>1,2</sup>
  - 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

---

<sup>1</sup>[https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl\\_parse.yrl](https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl)

<sup>2</sup>[http://www.erlang.org/doc/reference\\_manual/data\\_types.html](http://www.erlang.org/doc/reference_manual/data_types.html)

<sup>3</sup>Manually counted





## Evaluation

- Data types<sup>1,2</sup>
  - 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

---

<sup>1</sup>[https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl\\_parse.yrl](https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl)

<sup>2</sup>[http://www.erlang.org/doc/reference\\_manual/data\\_types.html](http://www.erlang.org/doc/reference_manual/data_types.html)

<sup>3</sup>Manually counted



## Evaluation

- Data types<sup>1,2</sup>
  - 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<sup>1,3</sup>:

---

<sup>1</sup>[https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl\\_parse.yrl](https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl)

<sup>2</sup>[http://www.erlang.org/doc/reference\\_manual/data\\_types.html](http://www.erlang.org/doc/reference_manual/data_types.html)

<sup>3</sup>Manually counted



## Evaluation

- Data types<sup>1,2</sup>
  - 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<sup>1,3</sup>:
  - Production rules: 255 (excluding deprecated features, also BNF is normalized)
  - Keywords: 34

---

<sup>1</sup>[https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl\\_parse.yrl](https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl)

<sup>2</sup>[http://www.erlang.org/doc/reference\\_manual/data\\_types.html](http://www.erlang.org/doc/reference_manual/data_types.html)

<sup>3</sup>Manually counted



## Evaluation

- Data types<sup>1,2</sup>
  - 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<sup>1,3</sup>:
  - 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)

---

<sup>1</sup>[https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl\\_parse.yrl](https://github.com/erlang/otp/blob/master/lib/stdlib/src/erl_parse.yrl)

<sup>2</sup>[http://www.erlang.org/doc/reference\\_manual/data\\_types.html](http://www.erlang.org/doc/reference_manual/data_types.html)

<sup>3</sup>Manually counted



## Evaluation

- Abstraction:

---

<sup>4</sup>[http://www.erlang.org/doc/reference\\_manual/modules.html](http://www.erlang.org/doc/reference_manual/modules.html)

<sup>5</sup>[http://www.erlang.org/doc/reference\\_manual/data\\_types.html](http://www.erlang.org/doc/reference_manual/data_types.html)

<sup>6</sup>[http://rosettacode.org/wiki/Scope\\_modifiers#Erlang](http://rosettacode.org/wiki/Scope_modifiers#Erlang)

<sup>7</sup><http://erlang.org/doc/man/erlang.html>



## Evaluation

- Abstraction:
  - **Modules and records** can be used to create data structures<sup>4,5</sup>

---

<sup>4</sup>[http://www.erlang.org/doc/reference\\_manual/modules.html](http://www.erlang.org/doc/reference_manual/modules.html)

<sup>5</sup>[http://www.erlang.org/doc/reference\\_manual/data\\_types.html](http://www.erlang.org/doc/reference_manual/data_types.html)

<sup>6</sup>[http://rosettacode.org/wiki/Scope\\_modifiers#Erlang](http://rosettacode.org/wiki/Scope_modifiers#Erlang)

<sup>7</sup><http://erlang.org/doc/man/erlang.html>



## Evaluation

- Abstraction:
  - **Modules and records** can be used to create data structures<sup>4,5</sup>
  - “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] **unless they are exported.**”<sup>6</sup>

---

<sup>4</sup>[http://www.erlang.org/doc/reference\\_manual/modules.html](http://www.erlang.org/doc/reference_manual/modules.html)

<sup>5</sup>[http://www.erlang.org/doc/reference\\_manual/data\\_types.html](http://www.erlang.org/doc/reference_manual/data_types.html)

<sup>6</sup>[http://rosettacode.org/wiki/Scope\\_modifiers#Erlang](http://rosettacode.org/wiki/Scope_modifiers#Erlang)

<sup>7</sup><http://erlang.org/doc/man/erlang.html>



## Evaluation

- Abstraction:
  - **Modules and records** can be used to create data structures<sup>4,5</sup>
  - “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] unless they are exported.**”<sup>6</sup>
- Expressivity: Has a rich set of functions in the core module, most pertinent are **optimization functions for handling data structures and concurrency**<sup>7</sup>

---

<sup>4</sup>[http://www.erlang.org/doc/reference\\_manual/modules.html](http://www.erlang.org/doc/reference_manual/modules.html)

<sup>5</sup>[http://www.erlang.org/doc/reference\\_manual/data\\_types.html](http://www.erlang.org/doc/reference_manual/data_types.html)

<sup>6</sup>[http://rosettacode.org/wiki/Scope\\_modifiers#Erlang](http://rosettacode.org/wiki/Scope_modifiers#Erlang)

<sup>7</sup><http://erlang.org/doc/man/erlang.html>





## Evaluation

- Exception handling:<sup>8</sup>

---

<sup>8</sup>[http://erlang.org/doc/reference\\_manual/errors.html](http://erlang.org/doc/reference_manual/errors.html)

<sup>9</sup><http://erlang.org/pipermail/erlang-questions/2013-March/072760.html>



## Evaluation

- Exception handling:<sup>8</sup>
  - Has a `try` and `catch` statements to handle exceptions in code

---

<sup>8</sup>[http://erlang.org/doc/reference\\_manual/errors.html](http://erlang.org/doc/reference_manual/errors.html)

<sup>9</sup><http://erlang.org/pipermail/erlang-questions/2013-March/072760.html>



## Evaluation

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

---

<sup>8</sup>[http://erlang.org/doc/reference\\_manual/errors.html](http://erlang.org/doc/reference_manual/errors.html)

<sup>9</sup><http://erlang.org/pipermail/erlang-questions/2013-March/072760.html>



## Evaluation

- Exception handling:<sup>8</sup>
  - 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**.”<sup>9</sup>

---

<sup>8</sup>[http://erlang.org/doc/reference\\_manual/errors.html](http://erlang.org/doc/reference_manual/errors.html)

<sup>9</sup><http://erlang.org/pipermail/erlang-questions/2013-March/072760.html>



## Evaluation

- Efficiency<sup>10,11,12</sup>

---

<sup>10</sup>[http://www.erlang.org/doc/efficiency\\_guide/myths.html](http://www.erlang.org/doc/efficiency_guide/myths.html)

<sup>11</sup>[http://www.erlang.org/doc/efficiency\\_guide/commoncaveats.html](http://www.erlang.org/doc/efficiency_guide/commoncaveats.html)

<sup>12</sup><http://learnyousomeerlang.com/types-or-lack-thereof>



## Evaluation

- Efficiency <sup>10,11,12</sup>
  - Employs **dynamic** type-checking

---

<sup>10</sup>[http://www.erlang.org/doc/efficiency\\_guide/myths.html](http://www.erlang.org/doc/efficiency_guide/myths.html)

<sup>11</sup>[http://www.erlang.org/doc/efficiency\\_guide/commoncaveats.html](http://www.erlang.org/doc/efficiency_guide/commoncaveats.html)

<sup>12</sup><http://learnyousomeerlang.com/types-or-lack-thereof>



## Evaluation

- Efficiency <sup>10,11,12</sup>
  - Employs **dynamic** type-checking
  - Erlang is an **interpreted language**

---

<sup>10</sup>[http://www.erlang.org/doc/efficiency\\_guide/myths.html](http://www.erlang.org/doc/efficiency_guide/myths.html)

<sup>11</sup>[http://www.erlang.org/doc/efficiency\\_guide/commoncaveats.html](http://www.erlang.org/doc/efficiency_guide/commoncaveats.html)

<sup>12</sup><http://learnyousomeerlang.com/types-or-lack-thereof>



## Evaluation

- Efficiency <sup>10,11,12</sup>
  - Employs **dynamic** type-checking
  - Erlang is an **interpreted language**
  - **Anonymous functions** (**fun**s) are notoriously slow, although improved in subsequent versions

---

<sup>10</sup>[http://www.erlang.org/doc/efficiency\\_guide/myths.html](http://www.erlang.org/doc/efficiency_guide/myths.html)

<sup>11</sup>[http://www.erlang.org/doc/efficiency\\_guide/commoncaveats.html](http://www.erlang.org/doc/efficiency_guide/commoncaveats.html)

<sup>12</sup><http://learnyousomeerlang.com/types-or-lack-thereof>





## Evaluation

- Efficiency<sup>10,11,12</sup>
  - Employs **dynamic** type-checking
  - Erlang is an **interpreted language**
  - **Anonymous functions** (**fun**s) are notoriously slow, although improved in subsequent versions
  - **Tail-recursive** functions are inherently faster than recursive ones.



---

<sup>10</sup>[http://www.erlang.org/doc/efficiency\\_guide/myths.html](http://www.erlang.org/doc/efficiency_guide/myths.html)

<sup>11</sup>[http://www.erlang.org/doc/efficiency\\_guide/commoncaveats.html](http://www.erlang.org/doc/efficiency_guide/commoncaveats.html)

<sup>12</sup><http://learnyousomeerlang.com/types-or-lack-thereof>

## Evaluation

- Efficiency <sup>10,11,12</sup>
  - Employs **dynamic** type-checking
  - Erlang is an **interpreted language**
  - **Anonymous functions** (**fun**s) 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.**”



---

<sup>10</sup>[http://www.erlang.org/doc/efficiency\\_guide/myths.html](http://www.erlang.org/doc/efficiency_guide/myths.html)

<sup>11</sup>[http://www.erlang.org/doc/efficiency\\_guide/commoncaveats.html](http://www.erlang.org/doc/efficiency_guide/commoncaveats.html)

<sup>12</sup><http://learnyousomeerlang.com/types-or-lack-thereof>

## Evaluation

- Efficiency <sup>10,11,12</sup>
  - Employs **dynamic** type-checking
  - Erlang is an **interpreted language**
  - **Anonymous functions** (**fun**s) 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...**”

---

<sup>10</sup>[http://www.erlang.org/doc/efficiency\\_guide/myths.html](http://www.erlang.org/doc/efficiency_guide/myths.html)

<sup>11</sup>[http://www.erlang.org/doc/efficiency\\_guide/commoncaveats.html](http://www.erlang.org/doc/efficiency_guide/commoncaveats.html)

<sup>12</sup><http://learnyousomeerlang.com/types-or-lack-thereof>



## Evaluation

- Efficiency <sup>10,11,12</sup>
  - Employs **dynamic** type-checking
  - Erlang is an **interpreted language**
  - **Anonymous functions** (**fun**s) 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.”

---

<sup>10</sup>[http://www.erlang.org/doc/efficiency\\_guide/myths.html](http://www.erlang.org/doc/efficiency_guide/myths.html)

<sup>11</sup>[http://www.erlang.org/doc/efficiency\\_guide/commoncaveats.html](http://www.erlang.org/doc/efficiency_guide/commoncaveats.html)

<sup>12</sup><http://learnyousomeerlang.com/types-or-lack-thereof>



## Evaluation

- Efficiency <sup>10,11,12</sup>
  - Employs **dynamic** type-checking
  - Erlang is an **interpreted language**
  - **Anonymous functions** (**fun**s) 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.”

---

<sup>10</sup>[http://www.erlang.org/doc/efficiency\\_guide/myths.html](http://www.erlang.org/doc/efficiency_guide/myths.html)

<sup>11</sup>[http://www.erlang.org/doc/efficiency\\_guide/commoncaveats.html](http://www.erlang.org/doc/efficiency_guide/commoncaveats.html)

<sup>12</sup><http://learnyousomeerlang.com/types-or-lack-thereof>



# Concurrent Programming Languages

Concurrent Programming Languages

Erlang

Limbo

Orc



## Background



---

<sup>13</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/limbo](http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo)

<sup>14</sup><http://www.vitanuova.com/inferno/index.html>

<sup>15</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/descent](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.”<sup>13</sup>



---

<sup>13</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/limbo](http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo)

<sup>14</sup><http://www.vitanuova.com/inferno/index.html>

<sup>15</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/descent](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.”<sup>13</sup>
- “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”<sup>14</sup>

---

<sup>13</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/limbo](http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo)

<sup>14</sup><http://www.vitanuova.com/inferno/index.html>

<sup>15</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/descent](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.”<sup>13</sup>
- “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”<sup>14</sup>
- Used to develop the **Inferno** operating system, which is meant for building cross-platform distributed systems<sup>14</sup>

---

<sup>13</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/limbo](http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo)

<sup>14</sup><http://www.vitanuova.com/inferno/index.html>

<sup>15</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/descent](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.”<sup>13</sup>
- “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”<sup>14</sup>
- Used to develop the **Inferno** operating system, which is meant for building cross-platform distributed systems<sup>14</sup>
- Built on top of **C**, and adopts a couple of C functionalities<sup>15</sup>



---

<sup>13</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/limbo](http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo)

<sup>14</sup><http://www.vitanuova.com/inferno/index.html>

<sup>15</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/descent](http://doc.cat-v.org/inferno/4th_edition/limbo_language/descent)

## Evaluation

- Data types<sup>16</sup>

---

<sup>16</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/limbo](http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo)

<sup>17</sup>Manually counted

<sup>18</sup><https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default>

<sup>19</sup><http://www.vitanuova.com/inferno/limbo.html>



## Evaluation

- Data types<sup>16</sup>
  - Integers, 64-bit long floating point real numbers, and strings as primitive types

---

<sup>16</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/limbo](http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo)

<sup>17</sup>Manually counted

<sup>18</sup><https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default>

<sup>19</sup><http://www.vitanuova.com/inferno/limbo.html>



## Evaluation

- Data types<sup>16</sup>
  - Integers, 64-bit long floating point real numbers, and strings as primitive types
  - Arrays, tuples, and lists as derived types

---

<sup>16</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/limbo](http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo)

<sup>17</sup>Manually counted

<sup>18</sup><https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default>

<sup>19</sup><http://www.vitanuova.com/inferno/limbo.html>



## Evaluation

- Data types<sup>16</sup>
  - 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”

---

<sup>16</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/limbo](http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo)

<sup>17</sup>Manually counted

<sup>18</sup><https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default>

<sup>19</sup><http://www.vitanuova.com/inferno/limbo.html>



## Evaluation

- Data types<sup>16</sup>
  - 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:

---

<sup>16</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/limbo](http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo)

<sup>17</sup>Manually counted

<sup>18</sup><https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default>

<sup>19</sup><http://www.vitanuova.com/inferno/limbo.html>





## Evaluation

- Data types<sup>16</sup>
  - 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:<sup>16,17</sup>
  - Keywords: 46<sup>17,18</sup>

---

<sup>16</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/limbo](http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo)

<sup>17</sup>Manually counted

<sup>18</sup><https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default>

<sup>19</sup><http://www.vitanuova.com/inferno/limbo.html>



## Evaluation

- Data types<sup>16</sup>
  - 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:<sup>16,17</sup>
  - Keywords: 46<sup>17,18</sup>
  - “Syntactically similar to C...”<sup>19</sup>

---

<sup>16</sup>[http://doc.cat-v.org/inferno/4th\\_edition/limbo\\_language/limbo](http://doc.cat-v.org/inferno/4th_edition/limbo_language/limbo)

<sup>17</sup>Manually counted

<sup>18</sup><https://bitbucket.org/inferno-os/inferno-os/src/83c37ad129003670023ebe2f8c318627ee4e84b0/limbo/lex.c?at=default>

<sup>19</sup><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



## 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**



## 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



## 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



## 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**



## 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





# Evaluation



## Evaluation

- Restricted aliasing:



## Evaluation

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



## 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.”



## 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.”



## 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.”



## Evaluation

- Efficiency:



## Evaluation

- Efficiency:
  - Employs **static and dynamic** type checking.





## 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.**”



## 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**”



## 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.**”



# Concurrent Programming Languages

Concurrent Programming Languages

Erlang

Limbo

Orc



# Background



## Background

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



## 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



## 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**.





## 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**



## Evaluation

- Data types<sup>21</sup>

---

<sup>21</sup><https://orc.csres.utexas.edu/documentation/html/userguide/userguide.html>

<sup>22</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.syntax.EBNF.html>

<sup>23</sup>Manually counted

<sup>24</sup><http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf>



## Evaluation

- Data types<sup>21</sup>
  - Integer, string, logical, and signals as primitive data types

---

<sup>21</sup><https://orc.csres.utexas.edu/documentation/html/userguide/userguide.html>

<sup>22</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.syntax.EBNF.html>

<sup>23</sup>Manually counted

<sup>24</sup><http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf>



## Evaluation

- Data types<sup>21</sup>
  - Integer, string, logical, and signals as primitive data types
  - Tuples, lists, and patterns as derived types

---

<sup>21</sup><https://orc.csres.utexas.edu/documentation/html/userguide/userguide.html>

<sup>22</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.syntax.EBNF.html>

<sup>23</sup>Manually counted

<sup>24</sup><http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf>



## Evaluation

- Data types<sup>21</sup>
  - Integer, string, logical, and signals as primitive data types
  - Tuples, lists, and patterns as derived types
- Syntax design:

---

<sup>21</sup><https://orc.csres.utexas.edu/documentation/html/userguide/userguide.html>

<sup>22</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.syntax.EBNF.html>

<sup>23</sup>Manually counted

<sup>24</sup><http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf>



## Evaluation

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

---

<sup>21</sup><https://orc.csres.utexas.edu/documentation/html/userguide/userguide.html>

<sup>22</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.syntax.EBNF.html>

<sup>23</sup>Manually counted

<sup>24</sup><http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf>



## Evaluation

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

---

<sup>21</sup><https://orc.csres.utexas.edu/documentation/html/userguide/userguide.html>

<sup>22</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.syntax.EBNF.html>

<sup>23</sup>Manually counted

<sup>24</sup><http://orc.csres.utexas.edu/documentation/OrcReferenceCard.pdf>



## Evaluation

- Abstraction:

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





## Evaluation

- Abstraction:
  - Definition of classes, ala-OOPL supported<sup>25</sup>



---

<sup>25</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.declarations.defclasses.html>

<sup>26</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.sites.custom.html>

<sup>27</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.stdlib.html#ref.stdlib>

<sup>28</sup><https://orc.csres.utexas.edu/papers/OrcExceptionSemantics.pdf>

<sup>29</sup><https://orc.csres.utexas.edu/documentation/html/refmanual/index.html>

## Evaluation

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

---

<sup>25</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.declarations.defclasses.html>

<sup>26</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.sites.custom.html>

<sup>27</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.stdlib.html#ref.stdlib>

<sup>28</sup><https://orc.csres.utexas.edu/papers/OrcExceptionSemantics.pdf>

<sup>29</sup><https://orc.csres.utexas.edu/documentation/html/refmanual/index.html>



## Evaluation

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



---

<sup>25</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.declarations.defclasses.html>

<sup>26</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.sites.custom.html>

<sup>27</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.stdlib.html#ref.stdlib>

<sup>28</sup><https://orc.csres.utexas.edu/papers/OrcExceptionSemantics.pdf>

<sup>29</sup><https://orc.csres.utexas.edu/documentation/html/refmanual/index.html>

## Evaluation

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



<sup>25</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.declarations.defclasses.html>

<sup>26</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.sites.custom.html>

<sup>27</sup><http://orc.csres.utexas.edu/documentation/html/refmanual/ref.stdlib.html#ref.stdlib>

<sup>28</sup><https://orc.csres.utexas.edu/papers/OrcExceptionSemantics.pdf>

<sup>29</sup><https://orc.csres.utexas.edu/documentation/html/refmanual/index.html>

## Evaluation

- Restricted aliasing: Employs **pass-by-value** (in the same manner as Java)<sup>30</sup>



---

<sup>30</sup><https://orc.csres.utexas.edu/papers/OrcJSSM.pdf>

<sup>31</sup><https://orc.csres.utexas.edu/download.shtml>

## Evaluation

- Restricted aliasing: Employs **pass-by-value** (in the same manner as **Java**)<sup>30</sup>
- Efficiency<sup>31</sup>



---

<sup>30</sup><https://orc.csres.utexas.edu/papers/OrcJSSM.pdf>

<sup>31</sup><https://orc.csres.utexas.edu/download.shtml>

## Evaluation

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

---

<sup>30</sup><https://orc.csres.utexas.edu/papers/OrcJSSM.pdf>

<sup>31</sup><https://orc.csres.utexas.edu/download.shtml>



## Evaluation

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

---

<sup>30</sup><https://orc.csres.utexas.edu/papers/OrcJSSM.pdf>

<sup>31</sup><https://orc.csres.utexas.edu/download.shtml>





## Evaluation

- Restricted aliasing: Employs **pass-by-value** (in the same manner as Java)<sup>30</sup>
- Efficiency<sup>31</sup>
  - 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

---

<sup>30</sup><https://orc.csres.utexas.edu/papers/OrcJSSM.pdf>

<sup>31</sup><https://orc.csres.utexas.edu/download.shtml>



## Evaluation

- Restricted aliasing: Employs **pass-by-value** (in the same manner as Java)<sup>30</sup>
- Efficiency<sup>31</sup>
  - 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**

---

<sup>30</sup><https://orc.csres.utexas.edu/papers/OrcJSSM.pdf>

<sup>31</sup><https://orc.csres.utexas.edu/download.shtml>



**END OF SESSION 8**

