

Loglan'82

więcej niż język programowania obiektowego i rozproszonego

Loglan'82 jest językiem programowania obiektowego i rozproszonego, ma wiele cech, które czynią z niego narzędzie programowania lepsze od innych:

- Ma unikalny, tani i bezpieczny system zarządzania obiektami.
- Oprócz modułów klas (**class**) oferuje moduły współprogramów (**coroutine**) i procesów (**process**). Możesz więc tworzyć nie tylko obiekty klas, ale także *obiekty współprogramów* i *obiekty procesów*.
- Loglanowskie maszyny wirtualne mogą się łączyć (przez sieć) w wirtualny, wieloprocesorowy komputer loglanowski by wspólnie wykonywać program(y).
- Obiekty procesów mogą być alokowane na różnych węzłach sieci połączonych maszyn wirtualnych, bądź na tym samym komputerze. Oznacza to, że Loglan ma *jeden model* obliczeń współbieżnych i rozproszonych (mniej uczenia się).
- Loglan'82 oferuje własny, oryginalny i w pełni obiektowy protokół komunikacji pomiędzy obiektami wątków tzw. obce wołanie metod (*ang.* alien call).
- Każdy wątek może stworzyć swój system współprogramów i zarządzać nim.

Kto może skorzystać na Loglanie?

Mamy nadzieję, że informacje zawarte na tych stronach okażą się interesujące i przydatne dla:

- **Ambitnych programistów** -- język Loglan'82 oferuje garść konstrukcji i rozwiązań nieznanych w innych językach programowania :
 - Jeśli chcesz świadomie zarządzać pamięcią obiektów i uniknąć przy tym niebezpiecznego zjawiska *wiszacych referencji* to spróbuj instrukcji **kill**().
 - Jeśli oprogramowanie, które stworzysz ma służyć obliczeniom współbieżnym, lub rozproszonym lub jakiejś ich kombinacji to wybierz Loglan'82 z jego jednym modelem obejmującym wszystkie te rodzaje obliczeń, zredukuje to znacznie koszty opanowania programowania współbieżnego i/lub rozproszonego.
 - Na pewno warto zapoznać się z *protokołem obcego wołania* metod wątku A podczas wykonywania instrukcji wątku B, jest to oryginalny wynalazek znany jak dotąd tylko w Loglanie.
 - Jeśli masz zamiar zaprogramować obliczenia quasi-współbieżne to zapoznaj się z współprogramami (**coroutines**).
- **Nauczycieli** -- Loglan'82 jest dobrym wyborem:
 - jeśli chcesz przedstawić wszystkie narzędzia i metody programowania obiektowego unikając przy tym przechodzenia od jednego do kolejnego języka programowania dla zilustrowania kolejnych narzędzi programowania obiektowego.
 - jeśli chcesz to możesz programować w Loglanie w ramach większych projektów programistycznych obejmujących specyfikowanie struktur danych i algorytmów, implementację specyfikacji czyli programowanie i weryfikację poprawności modułów oprogramowania względem specyfikacji.
- **Badaczy** -- Ponieważ język programowania obiektowego, powstał w wyniku poszukiwania odpowiedzi na kilka problemów. Odpowiedzi na poniższe pytania stanowią wkład intelektualny w projekt Loglan'82.

- ✓ **P1:** Czy możliwe jest programowanie, usuwanie niepotrzebnych obiektów - *bezpieczne* tj. wolne od zjawiska wiszących referencji i przy tym niedrogie? zob. [bezpieczna dealokacja obiektów](#) .
- ✓ **P2:** W jaki sposób wyznaczyć klasę dziedziczną, znając jej nazwę? W programie loglanowskim lub Javowym, może pojawić się wiele klas o tej samej nazwie na różnych poziomach drzewa zagnieżdżonych modułów?
- ✓ **P3:** Gdzie szukać deklaracji identyfikatora *id*, który pojawia się w jakimś miejscu programu? Jest to pytanie o definicję statycznego wiązania wystąpień aplikacyjnych z odpowiednimi deklaracjami identyfikatorów.
- ✓ **P4:** Czy znany z Algolu'60 i Pascala mechanizm adresowania wielkości nielokalnych (tzw. mechanizm Display Vector'a Dijkstry) można zastosować w językach takich jak Loglan (i teraz Java)?
- ✓ **P5:** Jak zarządzać obiektami współprogramów by uniknąć sprzeczności jakie pojawiły się w innych językach programowania z coroutinami?
- ✓ **P6:** Jak ma przebiegać tworzenie i zarządzanie obiektami procesów?
- ✓ **P7:** Jakie narzędzia wybrać do komunikacji i synchronizacji obliczeń współbieżnych?
- ✓ **P8:** Jak zarządzać obliczeniami rozproszonymi?
- ✓ **P9:** Czy istnieje różnica pomiędzy obliczeniami współbieżnymi a równoległymi i rozproszonymi?

Wyniki uzyskane podczas prac nad Loglanem okazały się przydatne w analizie języka Java. Mogą znaleźć zastosowanie w innych językach.

Co możesz pobrać i użytkować?

- Raport języka, podręczniki, instrukcje użytkownika, kompilatory na platformy Linux i Windows, zestaw przykładowych programów.
- Raporty zawierające rozwiązania wielu problemów wymienionych powyżej, czyli wcześniejsze wersje publikacji.

Ile to kosztuje? Ile na tym można zarobić?

Wszystko udostępniamy na licencji otwartego oprogramowania i otwartej nauki. Nie płacisz nic.

Spróbujmy oszacować zyski jakie można uzyskać:

Gdyby najczęściej stosowane języki programowania Java i C++ zaadaptowały system zarządzania obiektami wymyślony przez prof. Antoniego Kreczmara(1945 – 1996), to zyski możnaby liczyć w dziesiątkach milionów euro/rok.

Zechciej porównać.

Heap system	Model D (e.g. Loglan'82)	Model A (e.g. C++, Pascal)	Model B (e.g. Java 1995, Python)
Pre-	Certain object <i>o</i> is referenced by the <i>n</i> variables $x_1 = x_2 = \dots = x_n$, $1 \leq i \leq n$.		
Code	kill(x_i)	delete(x_i); $x_i = \text{null}$	$x_1 = \text{null}$; $x_2 = \text{null}$;... $x_n = \text{null}$; Now, the instruction gc() - the object <i>o</i> will be deleted.
Post-	All the variables took the value none. Object <i>o</i> is deleted.	Object <i>o</i> has been deleted. The variable x_i has the value null. Other variables point to the deleted frame.	Object <i>o</i> has been deleted – <i>under condition that all the strong</i> (normal) references to the object have been earlier assigned the null value.
Cost	O(1)	O(1)	O($n + m$) <i>m</i> is the global size of the heap of objects.
Risk	No risk(!) Each attempt to read and/or write from the deleted object, will raise an error signal <i>reference to none</i> .	If $n > 1$ then dangling reference error occurs. High probability of the error of contradicting information and/or destruction error.	Risk of memory leakage error , if programmer will forget to nullify some reference to the object <i>o</i> . It will remain not deleted.

To jest tylko odpowiedź na pytanie P1. Należy do tego dodać zyski, jakie można osiągnąć stosując inne wynalazki oferowane przez twórców Loglanu.

Dołącz do nas!

<mailto:a.salwicki@uksw.edu.pl>

Możesz pomóc w wielu pracach od drobnych, porządkowych do samodzielnego rozwiązywania problemów.

Wyniki naszych badań i nasze wynalazki znalazły

zastosowanie

w Loglanowskiej maszynie wirtualnej VLP:

- łączenie maszyn wirtualnych poprzez sieć w wirtualny wieloprocessorowy komputer loglanowski,
- tworzenie obiektów procesów,
Procesy są modułami programu. Obiekty procesów zachowują się odmiennie od obiektów klas: obiekty procesów mogą znajdować się w stanach Aktywny lub Pasywny. Obiekty aktywne działają współbieżnie, każdy na swoim procesorze.
- oryginalny protokół obcego wołania metod procesów,
- zarządzanie pamięcią obiektów,
- zarządzanie obiektami współprogramów
- szybkim dostępie do wielkości nielokalnych – wariant mechanizmu Display Vector specjalnie dopasowany do Loglanu z jego dwoma operacjami : zagnieżdżenia modułów i dziedziczenia klas.
- i in.

w kompilatorze Loglanu:

- algorytm identyfikacji klas dziedziczonych - *Inh*
- algorytm identyfikacji (*statycznego wiązania*) deklaracji identyfikatora odpowiedniej dla aplikacyjnego wystąpienia tego identyfikatora - *Bind*
- i in.

Oferujemy te wynalazki do zastosowania w bardziej popularnych językach programowania takich jak C++, Java i in.

Niech korzystają z naszego dorobku.

Na pierwszy ogień proponujemy przeprowadzenie następujących prac wdrożeniowych:

- w kompilatorze Javy wykorzystanie algorytmu LSW
oczekiwana korzyść: klarowna semantyka, poprawna diagnostyka błędów „cykl w relacji *dep* zależności.
- W maszynie wirtualnej Javy i w C++ zastosowanie systemu zarządzania obiektami z instrukcją *kill()* oraz sprawdzaniem czy obiekt żyje.
- Wykorzystanie protokołu *alien call* do celów komunikacji i synchronizacji obliczeń współbieżnych i rozproszonych
- Łączenie maszyn wirtualnych ...