



Opis dla każdego

Larch jest składającym się z **modułów** systemem służącym do **wspomagania dowodzenia**. W tym projekcie skupiamy się na poprawie metod **dydaktyki logiki** chcąc doprowadzić do możliwości **dowodzenia tylko i wyłącznie myszką**. Aktualnie wspieramy tabele analityczne KRZ (sygnowane i niesygnowane) oraz oferujemy konsolową implementację rachunku sekwentów.

The truth is only one click away!

Opis dla niektórych

Larch jest silnikiem wspomagania dowodzenia opartym na silnej implementacji Plugin Oriented Programming. Projekt rozwijany jest w celach dydaktycznych - aby odciążyć studentów od przepisywania zdań oraz rozpisywania reguł, dając tym samym możliwość skupienia się na strategii dowodu. Zaawansowanym użytkownikom oferowana jest możliwość tworzenia systemów dowodzenia, parserów, form wydruku, odpowiedzi, a nawet interfejsów.

Instalacja

Windows

1. Upewnij się, że masz zainstalowany co najmniej **Python 3.9**
2. Pobierz **release** (larch.zip) programu z [tego miejsca](#). Zalecamy wybór wersji GUI, gdyż uruchamia się ona domyślnie w formie aplikacji webowej.
3. Wypakuj zawartość.
4. Uruchom aplikację przez plik `.cmd`, lub `.pyz`.
5. Nalej sobie bezalkoholowego szampana, gdy Larch pobiera automatycznie wszystkie potrzebne pliki.

Mac oraz Linux

1. Upewnij się, że masz zainstalowany co najmniej **Python 3.9**
2. Pobierz **release** (larch.zip) programu z [tego miejsca](#). Zalecamy wybór wersji GUI, gdyż uruchamia się ona domyślnie w formie aplikacji webowej.
3. Wypakuj zawartość.
4. Uruchom plik `install.sh`. Nada on temu plikowi uprawnienia do bycia wykonywanym.
5. Program możesz uruchamiać wpisując w terminalu (w katalogu z plikami programu) `larch.sh`, bądź uruchamiając ten skrypt w dowolny inny sposób.
6. Nalej sobie bezalkoholowego szampana, gdy Larch pobiera automatycznie wszystkie potrzebne pliki.

Użycie i uruchamianie

Uruchamianie

Powtórz kroki od 5. w instrukcji instalacji. Wersja GUI dla bezpieczeństwa wymaga dostępu do internetu, aby przeprowadzać redownload wszystkich potrzebnych plików. W razie potrzeby możemy Ci jednak dostarczyć

wersję nie korzystającą z niego.

Użycie GUI

Interfejs jest dość prosty 😊 Kilka podpowiedzi:

1. Możesz zmieniać gałęzie klikając na ich liście.
2. Możesz rozkładać formuły w ramach różnych gałęzi przez kliknięcie na liść i wybór formuły do rozłożenia.
3. Musisz kliknąć regułę dowodzenia.

Użycie CLI

```
Logika -> Psychika
Type ? to get command list; type [command]? to get help
# prove ~((p -> (q -> r)) -> ((p v q) -> r)) Rozpoczęcie nowego dowodu
Sentence tokenized successfully
Proof initialized
Linen # use false imp 1 Oznaczenie aktualnej gałęzi oraz użycie reguły
Used 'false imp' successfully
No contradictions found on branch Linen. Mechanizm wykrywania sprzeczności (przy wykryciu zamknie gałąź)
Linen # use false imp 3
Used 'false imp' successfully
No contradictions found on branch Linen.
Linen # use true or 4
Used 'true or' successfully
No contradictions found on branch Linen.
No contradictions found on branch Turquoise.
Linen # jump > Skakanie między gałęziami
Branch changed to the right neighbour
Turquoise #
Program zawsze informuje Cię o aktualnej gałęzi
```

Aktualna gałąź

```
1. ~((p -> (q -> r)) -> ((pvq) -> r))
2. p -> (q -> r)
3. ~((pvq) -> r)
4. pvq
5. ~r
6. q
```

Najważniejsze komendy dostępne w interfejsie:

- `?` przywołuje listę wszystkich komend, `[komenda]?` przywołuje pomoc dla danej komendy,
- `prove [zdanie]` rozpoczyna dowód; możesz go opuścić z pomocą `leave`,
- w trakcie dowodzenia możesz używać komendy `use [reguła] [kontekst]` do użycia reguły,
- listę dostępnych reguł znajdziesz w podpowiedziach podczas wpisywania oraz pod komendą `get rules`, która podaje także informacje o wymaganiach danej reguły (na przykład poinformuje o potrzebie podania *Sentence ID*, czyli numeru zdania w gałęzi),
- **w każdym momencie dowodu znajdujesz się na pewnej gałęzi**, zmieniać możesz je komendą `jump [nazwa gałęzi]/>/<`, lub `next` (ta przeniesie Cię do następnej otwartej gałęzi),
- swój dowód możesz wyświetlić w łatwiejszej do zrozumienia formie z pomocą komendy `get tree`, gałąź możesz wyświetlić komendą `get branch`,

Larch oferuje też wewnętrzny system pluginów, ich listę możesz wyświetlić komendą `plugin list [nazwa gniazda/all]`. Pluginy możesz zmieniać komendą `plugin switch [gniazdo/aktualnie podłączony plugin] [nowy plugin]`.

Contributing

Zapraszamy do zapoznania się z naszą [dokumentacją](#) na ten temat!

Zgłaszanie błędów

Błędy (oraz propozycje) można zgłaszać za pomocą [Notion](#) oraz w [formularzu](#). Na powyższej stronie możesz znaleźć przycisk New. Powinna wyświetlić Ci się pusta strona z możliwością wyboru wzoru. Wybierz odpowiedni i wypełnij formularz. Postaraj się wyjaśnić jak najdokładniej, co się stało - każdy szczegół może okazać się przydatny! Jeśli masz pomysł, co mogło spowodować błąd, możesz spróbować samemu go naprawić!

Bardzo przydatne jest dla nas dołączenie crash reportu, który możesz znaleźć w folderze `crashes`. Chętnie przyjmujemy też plik `config.json`!

Tworzenie pluginów

Projekt Larch został utworzony tak, aby umożliwić każdej chętnej osobie tworzenie autorskich metod dowodzenia, formatów wydruku, formatów zapisów, czy interfejsów.






Możesz skopiować wzór pluginu z pomocą komendy `plugin gen [nazwa gniazda] [nazwa pluginu]`. Znajdziesz w nim zestaw wzorów funkcji, które wymagane są od danego pluginu. Type hinting podpowie Ci, co dany plugin przyjmuje, a co zwraca dana funkcja. Skorzystaj z docstringów wewnątrz kodu oraz [dokumentacji](#). Nie bój się z nami kontaktować - nie ma głupich pytań, są tylko głupie filmiki w internecie!

Dołącz do drzewnej drużyny

Larch jest projektem potrzebującym ludzi z szerokiego spektrum umiejętności. Potrzebujemy zarówno logików, programistów, testerów, jak i projektantów oraz grafików. W związku z tym chętnie przyjmujemy każdego do naszej społeczności - jeżeli uważasz, że możesz nam się przydać, to prawdopodobnie tak jest!

Jeśli chcesz się zaangażować, możesz skontaktować się z dowolnym z autorów.

Autorzy

- Jakub Dakowski (@PogromcaPapai) -  Benevolent dictator for life 
- Barbura Adamska - Chief Bzdury Officer 
- Robert Szymański (@rsxxi) - PM/UX unicorn :unicorn:
- Ola Draszewska (@nerdolo) - Confused coder 
- Łukasz Abramowicz (@ghostbuster265) - I arise from depths of my parents basement to code or sth 
- Dominika Juszcak (@antykwariat) - frontend develoops i did it again :lizard:

Oraz wszyscy, którzy bacznie przyglądali się rozwojowi aplikacji.

Autorzy pluginów

- Michał Gajdziszewski - algorytm printujący w `actual_tree`
- Aleksander Kiryk - algorytm generowania formuł