Chlang - 94^{404} *schackbottar*

Tage Danielsson

E-Mail: danielsson.dev@gmail.com
Skola: Hitachigymnasiet Västerås
Kurs: Gymnasiearbete
Datum: 22 april 2025

Innehåll

1 Sammanfattning

Chlang är ett språk som är anpassat för att skapa evalueringsfunktioner som tillsammans med en trädsökningsalgoritm bildar en schack-bot. En fil med innehållande chlang kan kompileras med hjälp av chlang kompilatorn för att skapa en sträng med 404 stycken specifika (mellan ascii värde 33 och 126) ascii tecken som sedan kan laddas in i något av chlang interfacen (webbsidan är lättast att använda), man kan då spela mot botten som representeras av strängen. Tanken med språket, som representeras av vilken (i princip) ascii sträng som helst med längden 404, är att man enkelt ska kunna dela bottar (då strängen kan fungera som ett bott id), generera bottar med algoritmer (då detta är lika enkelt som att skriva en algoritm som genererar strängen) och skapa bottar från berättelser (här finns ju tyvärr begränsningen av att till exempel å,ä och ö inte kan användas och att längden måste vara exakt 404, även mellanslag är otillåtet så man får i dessa fall använda till exempel understreck). Tanken är också att man ska kunna träna bottar med en genetisk algoritm, men detta går för tillfället långsamt och har inte gett några bra resultat. Man ska också kunna ta en sträng och köra kompilatorn baklänges för att få chlang igen men detta ingår fortfarande i sectionen vidare utveckling.

Chlang programmen och platformarna har utvecklats under läsåret 2024-2025. Samtliga är skrivna i programmeringspråket rust, men web-applikationen har även en html-fil. Flera verktyg, bibliotek och ramverk har använts under utvecklandet, dessa är:

Hyperfine - för prestandamätning

Cargo Flamegraph - för prestandamätning och visualisering

Rand - ett rust bibliotek för slumptalsgenerering.

Rustc-hash - ett rust bibliotek för snabbare hashfunktioner

Backtrace-on-stack-overflow - ett rust bibliotek för felsökning

Leptos - ett web ramverk för rust/wasm (för web applikation)

Gloo-timers - ett rust bibliotek för att hantera timers (som futures) i wasm.

Pix-engine - en game-engine för rust (för windows gui)

2 Abstract

Chlang is a language created for the configuration of custom evaluation functions that, together with a tree search, makes for a chess bot. A file containing valid chlang can be compiled using the chlang compiler to create a 404-character long string containing ascii characters between the values 33 and 126. These can then be loaded by one of the chlang runtimes (The website is the most accessible one) after wich you can play the chess bot represented by the string. The idea is that it should be easy to share bots (since the string is essentially an id), generate bots using code (since it's as easy as generating a string) and create bots from stories (this is a little limited since characters from other languages, such as the swedish "åäö" can not be used and space is also prohibited and has to be substituted with for ex (since it's as easy as generating a string) and create bots from stories (this is a little limited since characters from other languages, such as the swedish "åäö", can not be used and space is also prohibited and has to be substituted with for example an underscore). In theory it should also make it easy to train bots using a genetic algorithm but in practice this have been slow and have not produced any good results. You

are also supposed to be able to decompile the string into chlang so that you can easily analyze generated/trained bots. This is still in the chapter "Vidare Utveckling" (Further Improvements).

The chlang programs and platform are all writen in the programming language rust during the academic year of 2024-2025. The web version also includes a single html file. A couple of tools, libraries and frameworks were used during the development, these are:

Hyperfine - for benchmarks

Cargo Flamegraph - for benchmarks, cpu-profiling and visualisation

Rand - a rust library for random number generation

Rustc-hash - a rust library for faster hashing functions

Backtrace-on-stack-overflow - a rust library for debugging

Leptos - a web framework for rust/wasm (used for web interface)

Gloo-timers - a rust library for handling timers and futures in wasm.

Pix-engine - a game engine for rust (used for windows gui)

3 Inledning

3.1 Syfte

Syftet med chlang är att skapa intresse för skapandet av schackbottar genom att göra det roligt, lättillgängligt och enkelt att dela.

3.2 Bakgrund

Det finns såklart redan andra schackbottar, det som gör chlang unikt är att så många olika strängar representerar en giltig schackbot. Det skapar möjligheter för att dela bottar, generera bottar, testa bottar och leka tanken av att skapa bottar som representeras av till exempel namn, text och/eller skämt.

3.3 Frågeställning

Går det att representera en schackbot med vilken sträng som helst och hur skulle dessa bottars beteenden variera?

4 Teori

5 Metod

5.1 Delar

För att skapa Chlang så krävdes utveckling av flera olika delar. Därför startades arbetet med planering av vilka delar som skulle ingå och i vilken ordning dessa skulle skapas. Listan har ändrats lite med arbetets gång men ser nu ut såhär:

1. Schackspel:
   (a) Datastruktur för representering av bräde
   (b) Generering av drag
   (c) huvudlöst schackspel (schack spels api)
2. Schackbotsmotor:
   (a) Trädsökningsalgoritm
   (b) Hårdkodad evalueringsfunktion
   (c) Datastruktur och metoder för slutgiltig evalueringsfunktion.
   (d) Trädsöksoptimisering (pruning, cache)
3. Chlang-språket
   (a) Konstruktion av evalueringsfunktion från sträng
   (b) Kompilator för Chlang
   (c) Dekompilator för Chlang (ej färdigt)
4. Platformar (interface)
   (a) Web
   (b) Terminal (inte så bra, mest för utveckling)
   (c) Gui (inte så bra, mest för utveckling)
5. Verktyg
   (a) Jämför bottar
   (b) Träna bottar

5.2 Datatyper

Programmeringspråket rust har ett mycket kraftfullt datatypssystem som vi utnytjar i chlang. De huvudsakliga beståndsdelarna av chlang är alla representerade som egna datatyper. Dessa inkluderar:

- Schackbrädet (struct)
- Sida [vit/svart] (enum)
- Position på brädet (struct)
- Typ av pjäs (enum)
- Partiets tillstånd (enum)
- bottar (struct)
- Spelare [bot/människa] (enum)

- Drag (struct)
- Evalueringfunktioner (struct)
- Nyckel för bräde (struct)
- Flera structs och enums för parsing i kompilatorn

5.3 Verktyg

På grund av att prestanda är en så viktig del av schack-bottar så blev två verktyg väldigt viktiga. Dessa är Hyperfine och Cargo Flamegraph. Dessutom så kraschade programmet med felmeddelandet ”Stack Buffer Overflow” vid en period under arbetet vilket kunde felsökas med hjälp av biblioteket ”backtrace-on-stack-overflow”.

Hyperfine användes för att göra prestandamätningar som kunde uttnyttjas för att avgöra om ändringar i projektet ledde till framsteg. En ändring gav en prestandaökning som med hjälp av hyperfine uppmättes till att ge ca 120 gånger högre prestanda.

Verktyget som förenklade processen av att hitta möjligheten för nämnda prestandaförbättring var ”cargo flamegraph”. Detta verktyg genererar så kallade ”flamegraphs” eller ”flamechart” genom att använda linux egna prestandamätare och cpu-profilerare ”perf” och sedan sammanställa datan i en pdf. Genom att kolla på dessa flamecharts upptäcktes möjligheten till prestanda förbättringen då grafen visade att samma frekventa anrop till funktionen ”Clone” tog upp en stor del av körningstiden. Förbättringen skedde genom att göra ändringar till koden som tillät utbytet av den kostsamma kloningen av schackbräden mot kloning av en mindre ”nyckel” som innehöll endast den information som faktiskt behövde klonas (det är denna struktur som kallas ”Nyckel för bräde” i listan av datatyper).

Felmeddelandet ”Stack Buffer Overflow” gör inte att man blir klokare på vad det är man gör fel. Möjligen skulle man kunna anta att man fastnat i en oändlig rekursion men det finns ingen möjlighet att avgöra i vilken funktion detta skulle ske. Biblioteket ”backtrace-on-stack-overflow” förenklade felsökningen i detta fall då den visade att det var funktionerna för att få kungens möjliga drag i fallen med eller utan möjlighet till rokad som anropade varandra fram och tillbaka på grund av ett logiskt fel i basfallet.

5.4 Schack API

5.5 Interface för evaluering

5.6 Webbsida

6 Resultat och analys

Det slutgiltiga resultatet består av ett språk, en kompilator, en websida och möjligheten till 94^{404}*schackbottar*.

6.1 Språket

Språket är ett configspråk för schackbottar. Mycket likt TOML eller json (det vore kanske smart att helt enkelt använda TOML eftersom att det då redan finns syntax highlighting m.m). Alla Chlang-filer ser ungefär likadana ut. Det finns sektioner, fält och värden.

6.1.1 Sektioner

Sektionerna är en av dessa:

- Pawn (bonde)
- Knight (riddare)
- Bishop (löpare)
- Rook (torn)
- Queen (drottning)
- King (kung)
- Extra (annat, endast rokader för tillfället)

6.1.2 Fält

Fälten beror på om sektionen är ”Extra” eller inte. För sektionen extra finns två fält: ”LongCastle” (värde för long rokad) och ”ShortCastle” (värde för kort rokad). För pjässektionerna (alla andra sektioner) finns 4 fält. Dessa är:

- Base (basvärde)
- Position (värde baserat på position)
- Attack (värde för motståndaren per inkommande attack)
- Moves (värde per möjligt drag)

6.1.3 Värden

Varje fält i varje sektion ska tilldelas ett värde. Värdet som ska tilldelas varje fält är:

LongCastle - ett tal mellan 0 och 94

ShortCastle - ett tal mellan 0 och 94

Base - ett tal mellan 0 och 94

Attack - ett tal mellan 0 och 94

Moves - ett tal mellan 0 och 94

Position - en matris med åtta rader där varje rad har åtta tal mellan 0 och 94 separerade med mellanrum

6.1.4 Exempel

Ett exempel på en giltig Chlang fil är:

6.2 Kompilatorn

6.3 Websidan

7 Disskusion

8 Slutsats

9 Vidare Utveckling

10 Källförteckning

11 Bilagor