

Fachbereich Mathematik und Informatik  
Arbeitsgruppe Künstliche Intelligenz  
Freie Universität Berlin

Studienarbeit

# Client-Server-Architektur und Testframework für das Spiel Go

Autor: Markus Decke  
Betreuer: Prof. Dr. Raul Rojas, Dr. Marco Block-Berlitz

Februar 2010

## **Zusammenfassung**

Diese Studienarbeit untersucht die notwendigen Ein- und Ausgaben für ein Computer-Go-Programm. Es wird das Protokoll Go-Text-Protokoll vorgestellt und Einsatzmöglichkeiten diskutiert. Dateiformate wie das Smart-Game-Format und Standard-Format werden untersucht. Es werden mit Hilfe dieser, Tests und Experimente durchgeführt und Ergebnisse diskutiert.

## **Danksagung**

Ich danke Prof. Dr. Rojas für die Annahme der Arbeit und seinen interessanten Vorlesungen während meines Studiums der Informatik an der FU Berlin. Ich danke Dr. Marco Block für die Betreuung meiner Studienarbeit und den Mitgliedern der AG Spieleprogrammierung.

Insbesondere danke ich meinem Onkel Craig Hutchinson für das Wecken meines Interesses an Go.

Zu guter letzt danke ich allen die meinen Text Korrektur gelesen haben und mit hilfreichen Formatierungstipps zur Seite standen.

# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>1</b>
1.1	Das Spiel Go . . . . .	1
1.2	Kifoo . . . . .	3
<b>2</b>	<b>Ein- und Ausgabe</b>	<b>5</b>
2.1	Netzwerk . . . . .	7
2.1.1	Go-Text-Protokoll (GTP) . . . . .	7
2.1.2	KGS Go Server . . . . .	8
2.2	Dateiformate . . . . .	9
2.2.1	Smart Game Format . . . . .	10
2.2.2	Standardformat . . . . .	11
2.2.3	XML-Formate . . . . .	12
2.2.4	Dateiformate im Vergleich . . . . .	13
<b>3</b>	<b>Implementation</b>	<b>17</b>
3.1	Netzwerk . . . . .	18
3.2	Dateiformate . . . . .	19
<b>4</b>	<b>Tests und Experimente</b>	<b>21</b>
4.1	Regressionstestumgebungen . . . . .	21
4.2	Testspiele auf KGS Server . . . . .	22
<b>5</b>	<b>Ausblick</b>	<b>26</b>

# Kapitel 1

## Einführung

In der vorliegenden Studienarbeit geht es um eine Erweiterung des Projektes Kifoo, das einen Computer befähigt, Go zu spielen. Go ist ein asiatisches Brettspiel von hoher Komplexität, noch komplexer als Schach [HMG97]. Das Spiel Go und dessen Regeln werden im Abschnitt "Das Spiel Go" erläutert. Eine Beschreibung des Projektes Kifoo findet sich im gleichnamigen Abschnitt. Die Erweiterungen von Kifoo beziehen sich auf die Ein- und Ausgabemöglichkeiten eines Computerprogramms, das Go spielt, ob nun auf einem lokalen Rechner oder im Internet.

### 1.1 Das Spiel Go

Go ist ein Jahrhunderte altes asiatisches Brettspiel und erfreut sich auch in Europa großer Beliebtheit. Das Ziel des Spiels ist es, möglichst viel Territorium auf dem Spielbrett zu besetzen, ob nun durch einfaches Besetzen mit den eigenen Steinen oder dadurch, dass man seine Steine so platziert, dass ein Territorium für den Gegner uninteressant wird.

Go besitzt wenige Regeln, die im Folgenden erläutert werden. Das Spielfeld besteht aus sich kreuzenden senkrechten und waagerechten Linien, deren Anzahl gleich ist. Es wird zumeist auf einem Spielbrett mit der Feldgröße  $19 \times 19$  (siehe Abbildung 1.1) oder auch  $9 \times 9$  gespielt. Das Spielbrett  $19 \times 19$  ist die

übliche Größe und besitzt 361 Schnittpunkte. Dies ist um ein Wesentliches größer als Schach mit 64 Feldern.

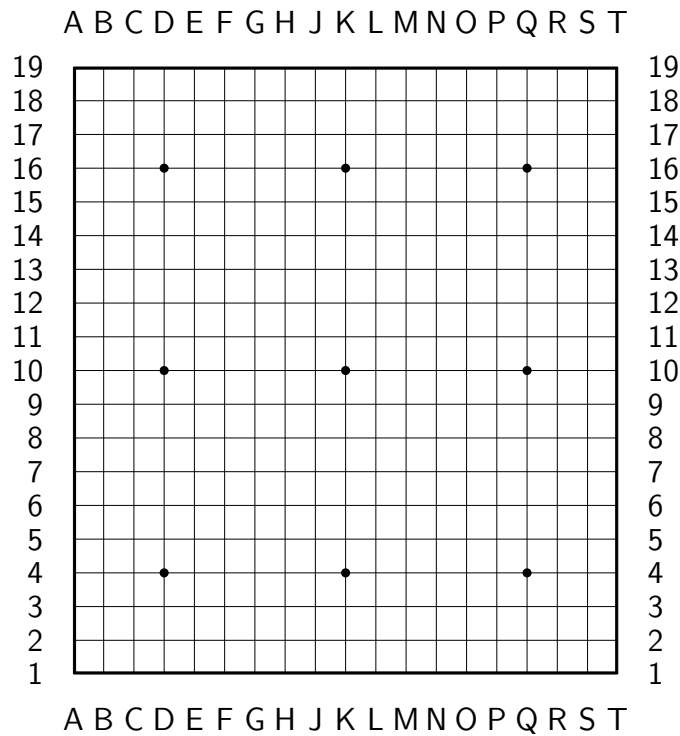


Abbildung 1.1: Spielbrettgröße 19

Es gibt zwei Spieler. Sie wechseln sich beim Setzen ihrer Steine ab. Die Steine werden durch die Farben Schwarz und Weiß unterschieden und dem jeweiligen Spieler zugeordnet. Steine können nur auf noch leere Punkte gesetzt werden. Die leeren, an einen Stein angrenzenden Punkte werden Freiheiten genannt. Diese können vom Gegner besetzt werden, um den dann umschlossenen Stein vom Spielbrett zu nehmen, also gefangen zu nehmen und so mehr Territorium oder Freiheiten zu erlangen. Die Aneinanderlegung von gleichfarbigen Spielsteinen führt nicht zum Verlust der Freiheit, sondern es werden der so entstandenen Kette alle leeren angrenzenden Punkte als Freiheit zugeordnet. Also muss man die gesamte Kette umschließen, damit man sie vom Spielbrett nehmen kann.

Es kann Situationen geben, in denen man sich gegenseitig immer wieder an gleichen Punkten Steine gefangen nimmt. Dies ist verboten, ein Spielbrett darf nicht in den vorhergehenden Zustand gebracht werden. Des weiteren ist ein sogenannter Selbstmord verboten, d.h. ein Stein darf nicht an einem Punkt abgelegt werden, an dem er keine Freiheiten mehr hat, also an einem Punkt, der vom Gegner komplett umschlossen ist, so dass das Setzen des Steines nicht zur Gefangennahme der gegnerischen Steine führt, sondern der eigenen. Das Spiel ist zu Ende, wenn beide Spieler hintereinander passen. Passen beide Spieler, so werden ihre Territorien verglichen. Der Spieler mit dem größten Territorium gewinnt.

Es gibt zwei unterschiedliche Zählweisen zum Ermitteln der Größe des Territoriums, einmal die chinesische und dann die japanische. Die chinesische Zählweise zählt alle Punkte, die man umschlossen hat und auf denen sich eigene Steine befinden. Die japanische Zählweise verwendet zusätzlich zum Territorium noch die gefangenen Steine, um das Endergebnis zu ermitteln. Die gefangenen Steine werden in das Territorium des Gegners gelegt, und es werden die verbliebenen freien Punkte gezählt.

Die Regeln des Spiels und der Auszählung sind ansprechend einfach und erlauben ein überaus komplexes Spiel. Informationen zum Spiel kann man beim Deutschen Go-Bund [DGO] im Internet finden.

## 1.2 Kifoo

Kifoo [kif09] ist ein studentisches Computer-Go-Projekt an der Freien Universität Berlin, für das Ein- und Ausgabe und auch eine Testumgebung erstellt werden sollen. Der Name Kifoo spielt auf den Begriff *kifuu* aus dem Japanischen an, was soviel wie Go-Spiel-Stil bedeutet. Im Moment ist der Spielstil des Projektes unbekannt und soll im Rahmen des Projekts noch gefunden werden. Testumgebungen und Spiele gegen Computer-Go-Programme und Menschen helfen bei der Suche



Abbildung 1.2: Kifoo Logo

Am Projekt Kifoo sind Daniel Wäber und Thorsten Reinhardt beteiligt. Ihr Blick geht in Richtung Evaluation von Heuristiken, Brettdarstellungen und Erweiterung der Monte-Carlo-Baumsuche. Es geht im Projekt auch ganz allgemein darum, ein Programm zu schreiben, das möglichst erweiterbar, benutzbar und intelligent ist, so dass eventuell auch andere Personen Lust haben, an dem Projekt mitzuarbeiten. Weitere Informationen bezüglich des Projekts stellt die Webseite des Projekts Kifoo bereit.

Für die Entwicklung des Spielstils werden Spielgegner benötigt. Diese kann man auf Internet-Servern finden. Auf diesen spielen Menschen und Computer-Go-Programme. Testweise kann man das Programm gegen sich selbst antreten lassen. Dies ist hilfreich, um zu sehen, ob eine Idee zu einem stärkeren Spiel führt, in Konkurrenz zu einer anderen Idee. Diese Vorgehensweise hat leider den Nachteil, dass wir das Spiel nicht allgemein bewerten können, da eventuelle Spielfehler und Reaktionen auf die Spielfehler implementationsabhängig sind. Es fehlt die Einschätzung gegenüber anderen Gegnern wie Computer-Go-Programme oder Menschen. Hierfür benötigen wir Ein- und Ausgabe-Möglichkeiten, so dass Kifoo mit Servern und anderen Programmen kommunizieren kann.

# Kapitel 2

## Ein- und Ausgabe

Dieser Abschnitt bietet einen Überblick über verschiedene Ein- und Ausgabemöglichkeiten, die Go-Programme nützlich erweitern. Folgende vier mögliche Anbindungen eines Go-Programms sind relevant: eine Anbindung an eine graphische Benutzeroberfläche (GUI), an einen Go-Server, an eine Go-Engine oder eine Testumgebung. Alle diese Fälle können mit einer Lösung abgedeckt werden, das Go-Text-Protokoll (GTP) [Far02]. Es kann verwendet werden, um sich mit einem im Internet befindlichen Server zu verbinden oder auch direkt mit anderen Go-Programmen. Es gibt weitere Protokolle, die GTP um eigene Kommandos erweitern, wie zum Beispiel der KGS-Go-Server. Einige Protokolle besitzen keinen genügend großen Mehrwert, es im Moment zu implementieren, wie das des Internet Go Server (IGS) [igs].

Es gibt verschiedene Möglichkeiten, Spiele auszugeben und zu speichern. Man kann die GTP-Ausgabe mitschneiden und so bearbeiten, dass man sie wieder einlesen kann, oder man kann direkt ein Dateiformat wählen, das auch von anderen Programmen genutzt werden kann, ohne weiteres Eingreifen per Hand. Mögliche Ausgaben sind das Smart-Game-Format [SGFc], Ishi bzw. Standard-Format [Fot90], ein XML-basiertes Format [XGF02] [Groat] oder auch ein Latex-Text-Format [Bos] [Aug].

Im Allgemeinen soll ein Dateiformat es ermöglichen, ein Spiel mit Kom-

mentaren und anderen Anmerkungen zu speichern und es bei Gelegenheit wieder einzulesen, um das Spiel zu bewerten oder nachzuvollziehen. Bei den Ein- und Ausgaben geht es generell darum, ob denn das Computer-Go-Programm, welches man geschrieben hat, sinnvoll spielt. Man kann dies leichter erkennen, wenn man eine übliche Go-Darstellung hat. Spiele nachzuvollziehen ist beim automatisierten Spielen auf Servern im Internet eine mühsame Aufgabe, da man nicht permanent zuschauen kann, um zu erkennen, ob ein Gegner wirklich verloren hat oder einfach nur keine Lust mehr hatte. Daher ist es nützlich, nachträglich zu erkennen, ob sinnvoll gespielt wurde oder nicht. Schwerwiegende Spielfehler eines Go-Programms führen schnell dazu, dass ein menschlicher Gegner die Lust an einem Spiel verliert und so ein Computer-Go-Programm ungerechtfertigt gewinnt. Diese Spiele sollen nach Möglichkeit nicht in die Bewertung des Spielstils des Computer-Go-Programms einfließen.

Es ist äußerst aufwendig, alle jemals gespielten Spiele durchzulesen, um zu erkennen, ob denn eine Verbesserung eingetreten ist. Dazu würde es auch nötig sein, das man selbst im Spiel besser wird. Da dies zu viel Zeit kostet, wird eine automatisierte Möglichkeit, Kifoo Spiel einzuschätzen, benötigt. Mit automatisierten und standardisierten Tests, auch Regressionstests genannt, ist es einfacher und schneller möglich, eine Bewertung des Spielstils vorzunehmen. Hierfür benötigen wir Spiele, deren bester Zug oder eine Auswahl von verschiedenen guten Zügen, bekannt ist. Die Spiele sollten im Schwierigkeitsgrad und im Spielverlauf variieren, um zu erkennen, ob Kifoo besser oder schlechter wird.

Eine Eingabemöglichkeit, die auch für Menschen leicht lesbar ist, wird benötigt. Es kommen SGF und andere Dateiformate in Frage, diese werden im Abschnitt "Dateiformate" diskutiert. Grundsätzlich benötigt man Editoren, um die Testdaten selbst zu erzeugen. Der KGS-Go-Server [KGSb] bietet die Möglichkeit, gespielte Partien runterzuladen. So kann man von anderen Spielern gute Partien zum Testen verwenden oder auch eigene Spiele und diese für Tests bewerten. Diese gewählten Partien sind dann einer Testumgebung

einzufragen, um die Implementation, Entwicklung und Evaluation von Kifos Spiel zu unterstützen.

## 2.1 Netzwerk

Um nicht nur lokal spielen zu können, wird eine Möglichkeit benötigt, sich mit anderen Programmen im Netzwerk auszutauschen. Es gibt hierfür das Go-Text-Protokoll (GTP) welches einen de-facto Standard darstellt, da viele Server und Programme nur das GTP für die Kommunikation mit anderen Programmen unterstützen, zum Beispiel KGS-Go-Server [KGSb] und Gnu-Go [Fou]. Es gibt Anbieter von Go-Servern, die kleinere Erweiterungen verwenden, z. B. eine Authentifizierung, die sie in eigenen GTP-Klienten anbieten, so dass man dafür selbst keine Änderung an seinem Computer-Go-Programm vornehmen muss. Über die Standard-Ein- und Ausgabe können Go-Programme, die GTP sprechen, auf dem gleichen Rechner gegeneinander antreten. Es gibt auch einige Programme, die eine graphische Oberfläche anbieten, an die man GTP-sprechende Programme anbinden kann, um sich deren Spiel visualisieren zu lassen. Ein solches Programm ist zum Beispiel Go-Gui [GoG] mit dem man ein Spiel anschauen oder auch spielen kann.

### 2.1.1 Go-Text-Protokoll (GTP)

Das GTP ist ein Netzwerk-Kommunikations-Protokoll für Computer-Go. Es ist nach Aussage des Entwerfers einfach zu implementieren, jede Nachricht, die übermittelt wird, erzeugt eine Antwort, die über Erfolg oder Misserfolg des Kommandos etwas aussagt. Auf diese Weise ist der Server über den Zustand des Programms informiert, z. B. auch, ob die Eingabe ausgeführt wurde oder nicht. Die Struktur der Kommandos ist in [Abbildung 2.1](#) zu sehen. Es wird eine persistente und direkte Kommunikation des Klienten mit dem Gegenüber verwendet. Dieser kann ein Server, Go-Programm oder auch eine GUI sein. Die Befehle besitzen Parameter, wie Spielfarbe und Koordinate des Zuges, und erwarten einen Rückgabewert. Es werden nur wenige Befehle benötigt, um mit GTP sprechenden Programmen zusammenzuarbeiten. Eine

```

Eingabe          [id] command_name [arguments]
Positive Antwort  =[id] result
Negative Antwort  ?[id] error_message

```

Abbildung 2.1: Struktur der GTP-Kommandos

Übersicht der allgemeinen Befehle findet sich in Abbildung 2.2. Es gibt noch

```

protocol_version  name  version  known_command
list_commands    quit  boardsize  clear_board
komi              play  genmove

```

Abbildung 2.2: GTP: allgemeine Kommandos

zwei optionale Gruppen von Befehlen, die implementiert werden können. Die Befehle in der Regressionsgruppe in Abbildung 2.3 sind für uns am relevantesten, da diese für Testumgebungen benötigt werden. Die Befehle bedeuten zum einen das Laden einer `.sgf` Datei und zum anderen das Ausführen eines Zuges für das geladene Spiel. Die Befehle aus Abbildungen 2.1 und 2.3 bieten

```

loadsgf  reg_genmove

```

Abbildung 2.3: GTP: Regressionskommandos

für uns die Basisfunktionalität, die wir benötigen, um Spiele gegen Kifoo zu spielen und um Kifoo zu testen. Die Befehle aus der Turniergruppe, siehe Abbildung 2.4, werden für Spiele gegen Menschen und Spiele gegen Programme, die Handicap benötigen oder auch zulassen, benötigt, um im Internet spielen zu können.

### 2.1.2 KGS Go Server

KGS Go Server (KGS) ist ein bekannter Server für Go und hat eine große Nutzergemeinschaft [KGSa]. Es gibt einen eigenen Computer-Go-Bereich in dem man seine Go-KI spielen lassen kann. Die Möglichkeit, gegen menschliche Spieler anzutreten, die einen Rang besitzen, ist sehr hilfreich, um die Stärke Kifoo einzuschätzen, da menschliche Spieler immer noch stärker spielen können als ein Computerprogramm [Wed] [HSU07]. KGS bietet einen

```
handicap free_handicap set_free_handicap
```

Abbildung 2.4: GTP: Turnierkommandos

Klienten [Ser], der ein GTP sprechendes Computer-Go-Programm an den Server anbindet. Zum Spielen auf KGS wird ein Benutzerzugang benötigt. Dieser ist kostenlos erhältlich. Die Authentisierung wird mit dem von KGS gelieferten Klienten erledigt, da GTP nicht über solche Befehle verfügt. KGS hat das GTP um eigene Befehle erweitert, zu finden in Tabelle 2.5. Diese werden benötigt, da man in GTP nicht ausdrücken kann, wieviele Steine in einer bestimmten Zeit noch spielbar sind (byo-yomi) und um das Spielfeld von toten Steinen zu räumen. Da diese kleinen Änderungen ausreichen, um auf KGS zu spielen, haben wir uns dafür entschieden, zumal es öffentlich zugänglich und dokumentiert ist.

```
kgs-genmove_cleanup kgs-time_settings
```

Abbildung 2.5: KGS GTP Erweiterungen

## 2.2 Dateiformate

Es folgt eine Diskussion einzelner Dateiformate, die genutzt werden können, um Go-Partien zu speichern. Es ist sinnvoll, gespielte Partien zu speichern, da man nicht alle Spiele, die automatisiert gespielt werden, in Echtzeit nachvollziehen und auswerten kann. Es sollte eine Möglichkeit geben, die Spiele nachträglich anzuschauen und eventuell Fehler in der Spielweise zu entdecken, da Spielfehler dazu führen können, dass ein Gegner genervt aufgibt, also nicht besiegt wurde.

Es gibt das etablierte Dateiformat Smart-Game-Format (SGF) [SGFc], das einen de-facto Standard zur Dokumentation von Go-Spielen darstellt. Es wurde entwickelt um verschiedene Brettspiele darstellen zu können. Es gibt weitere spezielle Go-Dateiformate Standard [Fot90] und Liberty [Lib]. Sie sind unabhängig voneinander entstanden und werden heute immer noch ge-

nutzt, meist aus historischen Gründen seitens der nationalen Go-Spieler-Vereinigungen. Die Verwendung von XML als Dokumentationsformat hat sich bislang nicht durchgesetzt. Gnu-Go und KGS müßten Aufgrund ihrer großen Benutzergemeinde das neue Format unterstützen, bevor es sich zu einer echten Alternative entwickeln könnte.

### 2.2.1 Smart Game Format

Smart-Game-Format (SGF) wurde von Anders Kierulf 1987 entwickelt und vorgestellt [Kie87]. Es ist bis zum Jahre 2006 weiterentwickelt worden [sgfd]. Aufgrund des weit verbreiteten Einsatzes des Dateiformats ist es zu einem Quasi-Standard erwachsen. Es ist ein Klartext-Dateiformat, und bietet verschiedene Auszeichnungsmöglichkeiten für Brettspiele im allgemeinen. Da es relativ alt ist, werden wenige Zeichen verwendet, um die Auszeichnungen zu konkretisieren, da es dann weniger Platz zum Speichern benötigt. Dadurch leidet die Lesbarkeit der Dateien. Ein Spielverlauf bis zu den ersten Zügen der jeweiligen Farbe und dem Spielanfang würde in SGF wie in [Abbildung 2.6](#) aussehen. Eine Partie besteht aus dem Spielbaum (Gametree), der mit

```
(  
;GM [1] FF [4] ST [1] SZ [9] HA [0] KM [5 . 5] PW [P2] PB [P1]  
;B [cc]  
;W [gg]  
;B [cg]  
;W [gc]  
;B [ee]  
)
```

Abbildung 2.6: Beispiel SGF

runden Klammern ( und ) umschlossen wird und Knoten (Nodes), die mit Semikolon beginnen und Eigenschaften (Properties). Diese werden mit maximal zwei Buchstaben bezeichnet und haben innerhalb der eckigen Klammern ihren Wert.

Bei dem Beispiel aus Abbildung 2.6 befindet sich `B[cg]` in der fünften Zeile im vierten Knoten. `B` bezeichnet die Farbe Schwarz, `cg` ist die Position. Es ist ein Zug der Farbe Schwarz auf dem Punkt `c7` auf einem Go-Spielbrett. Wegen der Verwendung von Groß- und Kleinschreibung in SGF, ist es möglich  $52 \times 52$  große Felder zu beschreiben. Leider wird in GTP, wie es auch bei Beschreibungen auf Papier Sitte ist, das `i` bzw. `I` ausgelassen, da es leicht zu Verwechslungen mit dem Buchstaben `j` bzw. `J` kommen kann. Diese Eigenart muss natürlich beachtet werden. In GTP ist die Groß- und Kleinschreibung nicht relevant, da man laut Spezifikation alle Eingaben in Kleinbuchstaben umwandeln sollte. D.h. man kann nur bis zu  $25 \times 25$  Spielbretter beschreiben, allerdings ist ein normal großes Brett  $19 \times 19$  groß. Das ist daher vollkommen ausreichend.

SGF bot, da es schon recht früh entwickelt wurde und zeichensparsam in den Auszeichnungen ist, eine gute Möglichkeit, Spiele auszutauschen, wenn nur eine schmale Internetverbindung vorhanden ist. Da es in Regressionstests verwendet wird, ist es für uns nötig, dies im Projekt zu verwenden. Es wäre sonst notwendig, schon existierende Tests umzuschreiben bzw. in ein anderes Dateiformat zu konvertieren. Diese Mehrarbeit erscheint nicht sinnvoll, da sehr viele Spiele in SGF vorliegen und auch bei KGS in diesem Format alle Spiele gespeichert und dokumentiert werden.

### 2.2.2 Standardformat

Das Standardformat, auch Ishi genannt, ist seit Frühjahr 1990 spezifiziert und wurde von David Fotland bereitgestellt [Fot90]. Es ist ein Klartext-Format und ist zeilen- und schlüsselwort-orientiert. Es ist linear aufgebaut und es stellt Zeile für Zeile einen Spielverlauf dar. Schlüsselwörter befinden sich am Anfang einer neuen Zeile. Ein kleines Beispiel findet sich in Abbildung 2.7. Es verbraucht im Vergleich zu SGF mehr Zeichen und damit Speicherplatz. Deshalb hat es sich evtl. nicht überall als Standardformat durchgesetzt. Es

```
BLACK Player1 ()
WHITE Player2 ()
HANDICAP 0
KOMI 5.5
B 1 c7
W 2 g3
B 3 c3
W 4 g7
B 5 e5
```

Abbildung 2.7: Beispiel ishi

ist für Menschen etwas besser lesbar als das SGF. Jede Aktion der Spieler wird nummeriert. So kann man den Spielzug an Hand seiner Nummer einordnen. Es unterstützt den Implementierer beim prüfen seiner Implementation, ob alle Spielzüge vom Computer-Go-Programm in der richtigen Reihenfolge ausgeführt bzw. gelesen werden.

### 2.2.3 XML-Formate

Es gab von den Entwicklern des SGFs Bestrebungen, ein XML basiertes Dateiformat XGF [XGF02] zu entwickeln. Es ist allerdings bisher nicht aus dem Stadium einer Diskussionsversion hinaus gekommen und demzufolge hat es sich nicht durchgesetzt [Nij06].

Viele Programme verwenden und exportieren SGF. Programmierer, die einen funktionierenden SGF-Leser geschrieben haben, haben keine Motivation, einen XML-Leser zusätzlich zu schreiben oder umzusteigen, wenn es kein häufig genutztes Format gibt. Einen XML-Leser verwenden zu können, würde die Entwicklung beim Lesen und Schreiben von Dateien etwas vereinfachen, da es in Programmiersprachen, wie z. B. Java, Python und C++, schon etablierte XML-Parser gibt, die lediglich das Schema lesen müssen und dann die Daten für das Programm bereitstellen können und evtl. entsprechende Klassen erstellen.

Einzelne Klienten verwenden ihr eigenes XML-Format und laden dazu ein dieses zu verwenden. Eines dieser XML-Formate stammt vom Programm JaGo [Grob], hier JaGo-XML genannt. Das JaGo-XML [Groa] verwendet faktisch die gleichen Auszeichnungselemente, die es in SGF schon gibt. D.h. es ändert sich prinzipiell sehr wenig für erfahrene SGF-Benutzer, besonders wenn sie einen SGF-Editor verwenden, wie z. B. von JaGo oder auch Go-GUI angeboten. Verwendung des jeweiligen Formats bleibt dem Anwender bei Benutzung einer grafischen Oberfläche verborgen.

Mithilfe einer XSL-Transformation kann JaGo-XML in HTML umgewandelt werden. Es würde tatsächlich Sinn ergeben, eine XSL-Transformation zu machen, um aus den Dateien auch SGF-Dateien zu erzeugen und so die Interoperabilität der Daten zu erhöhen. Im Moment müsste man JaGo starten und das JaGo-XML einlesen und es als SGF speichern. Da im Moment kein XML-Format zwingend nötig ist, da die meisten Partie-Sammlungen im `.sgf` und `.sf` Format sind, findet keines bei Kifoo Verwendung.

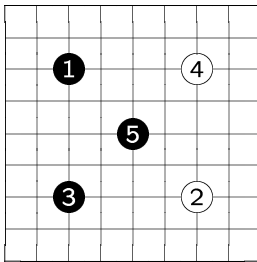
## 2.2.4 Dateiformate im Vergleich

Es existieren mehrere Dateiformate für die Speicherung von Go-Spielen auf dem Computer. Das gebräuchteste ist SGF, es existieren daneben das sogenannte Standardformat (Ishi) und das Liberty-Format. Diese beiden anderen Formate hatten sich parallel zum SGF entwickelt und hatten zum Teil unterschiedliches Publikum. Das Libertyformat ist für ein Programm entwickelt worden, das Go-Spiele auf dem Atari ST bearbeitet. Es ist von Jan van Steen entwickelt worden, es hat keine große Verbreitung erfahren.

Das Standard-Format könnte innerhalb eines Computer-Go-Programms zur Protokollierung von Spielen verwendet werden. Leider ist es durch seine aktuell geringe Verbreitung nicht erstrebenswert, es in Kifoo mit einzubinden. Es existiert ein Programm, Liberty- und Standardformat in SGF umzuwandeln, `sgf2misc` [sgfb] erleichtert die Portierung von interessanten Spielen, die nicht in SGF vorliegen. Mit Hilfe von `sgf2dg` kann man Ansichten für Latex erzeugen.

gen [Aug], zu sehen ist eines in Abbildung 2.8. Leider ist beim einbinden des Diagramms etwas Handarbeit notwendig. Eine Alternative stellt `psgo` [Bos] dar.

**White:** Player2  
**Black:** Player1  
**Komi:** 5.5



**Diagram 1:** 1-5

Abbildung 2.8: `sgf2dg` Ausgabe des SGF Beispiels

Wenn SGF als einziges Format genutzt wird, muss beachtet werden, dass es zwar eine Spezifikation des SGF gibt, sich aber nicht alle daran halten. Manche Programme haben SGF durch eigene Auszeichnungselemente ergänzt. Dadurch kann es zu Export- und Import-Inkompatibilität kommen. Nur die Ergänzungen können sich durchsetzen, die von einer großen Zahl von Menschen verwendet und in den wichtigsten Programmen implementiert werden. Ein Programm kann zwar Erweiterungen, die über das normale SGF hinausgehen, verwenden. Aber nur ein Programm, das diese auch verwendet, kann dies anzeigen oder verarbeiten. Problematisch sind zusätzliche Auszeichnungselemente also Erweiterungen, die von Programm zu Programm semantisch unterschiedlich bewertet werden.

XML-Formate werden sich erst durchsetzen, wenn sich beliebte Server oder Programme dazu entschließen würden, XML-Formate zu unterstützen. Im Moment sieht es nicht danach aus. Die meisten Programme sind etabliert und stabil, viele Entwickler sehen keinen zusätzlichen Nutzen ein XML-Format zu unterstützen.

SGF ist für Menschen nicht gut lesbar [sgfa]. Das Standard-Format ist für Menschen recht gut lesbar. Es benötigt im Gegensatz zu XML, wenig Zeichen, um die Struktur, die nötig ist, zu erzeugen. Deren Daten bestehen fast nur aus Struktur-Anweisungen, was die Lesbarkeit verringert. Es ist sinnvoll, ein Dateiformat zu wählen, das sich leicht in ein anderes überführen lässt und dessen Ansicht für Menschen gut lesbar ist. Die Partie aus Abbildung 2.9 ist eine bildliche Ausgabe des Beispiels SGF aus Abbildung 2.6. Es wurde mit `psgo` [Bos] für Latex erzeugt. Diese Darstellung eignet sich sehr gut, um

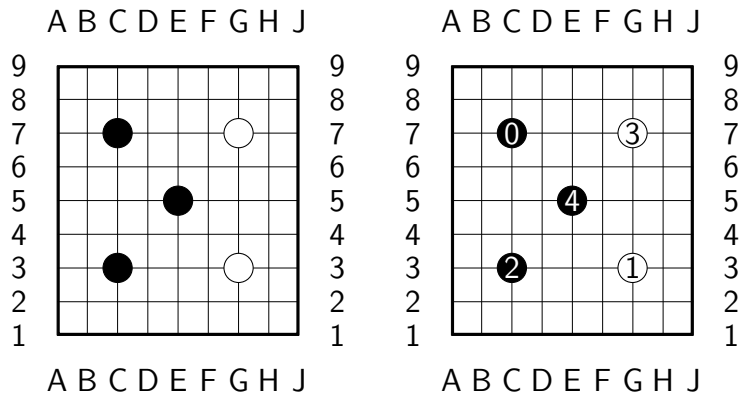


Abbildung 2.9: Beispiel Ausgaben

Partien für Menschen lesbar zu machen, entweder als Klartext-Datei oder als `.pdf`-Datei, die in SGF-Dateien vorliegen, deshalb wurde eine `psgo`-Ausgabe in Kifoo integriert.

Das Programm `sgf2dg` existiert zwar für Latex-Abbildungen. Eine solche Ausgabe ist als Klartext-Datei sehr schlecht lesbar im Gegensatz zu `psgo`, siehe Abbildung 2.10. Wegen der guten Unterstützung von SGF in anderen Programmen und Servern wird es für Kifoo verwendet.

```

\setbox\diagbox=\vbox to 120.0 pt{
\hsize= 108.0 pt\goo
\0??<\0??(\0??(\0??(\0??(\0??(\0??(\0??>
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\001+\0??+\0??+\0??+\004+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\005+\0??+\0??+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??[\0??+\003+\0??+\0??+\0??+\002+\0??+\0??]
\0??[\0??+\0??+\0??+\0??+\0??+\0??+\0??+\0??]
\0??,\0??)\0??)\0??)\0??)\0??)\0??)\0??)\0??}
\vfll}

```

Listing 2.1: Ausschnitt des Textinhalts von 2.8

```

\begin{psgoboard}[9]
\move{c}{7}
\move{g}{3}
\move{c}{3}
\move{g}{7}
\move{e}{5}
\end{psgoboard}

```

Listing 2.2: Vollständiger Textinhalt von 2.9

Abbildung 2.10: Darstellungen in Latex-Quelltext

# Kapitel 3

## Implementation

Dieser Abschnitt beschreibt die Implementation der Netzwerkschnittstelle und Dateischnittstelle, die für Kifoo verwendet wurde. Kifoo wird in der Java Virtual Machine (JVM) ausgeführt. Die Erweiterungen Kifoos wurden deswegen in Java geschrieben. Die Idee der Implementation der Protokolle in einer einzigen Klasse, die alle Befehle und Daten verarbeitet, wurde schnell verworfen. Es führte zu Problemen in der Berücksichtigung verschiedener kleiner Erweiterungen von GTP. Diese Erweiterungen, die von einigen Entwicklern in ihren Protokollen verwendet werden, sind geeignet, in einer eigenen Klasse Verwendung zu finden. Mit Hilfe von Vererbung, die Java bietet, lassen sich die Erweiterungen in Unterklassen abbilden. Es müssen nur neue Befehle implementiert werden. Schon implementierte Befehle der Oberklasse lassen sich mit `super().action(cs)` aufrufen. Dies erleichtert die Implementation und lässt ohne weiteres verschiedene Implementationen einzelner Befehle für verschiedene Server oder Programme zu. Solange ein Protokoll auf GTP basiert, ist es sehr leicht, Weiteres in Kifoo hinzuzufügen. Für Kifoo wurde eine Eingabe von SGF-Daten implementiert, sowie die Ausgabe von SGF und `psgo`-Text.

## 3.1 Netzwerk

Die Netzwerkschnittstelle wurde in einem eigenen Thread implementiert, um so auf Ereignisse zu reagieren und diese zu verarbeiten, während Kifoo denkt. Die Weiterleitung der eingehenden Nachrichten ist in [Abbildung 3.1](#) graphisch als Sequenzdiagramm dargestellt.

Eingehende Nachrichten werden an den Handler weitergereicht. Dieser gibt

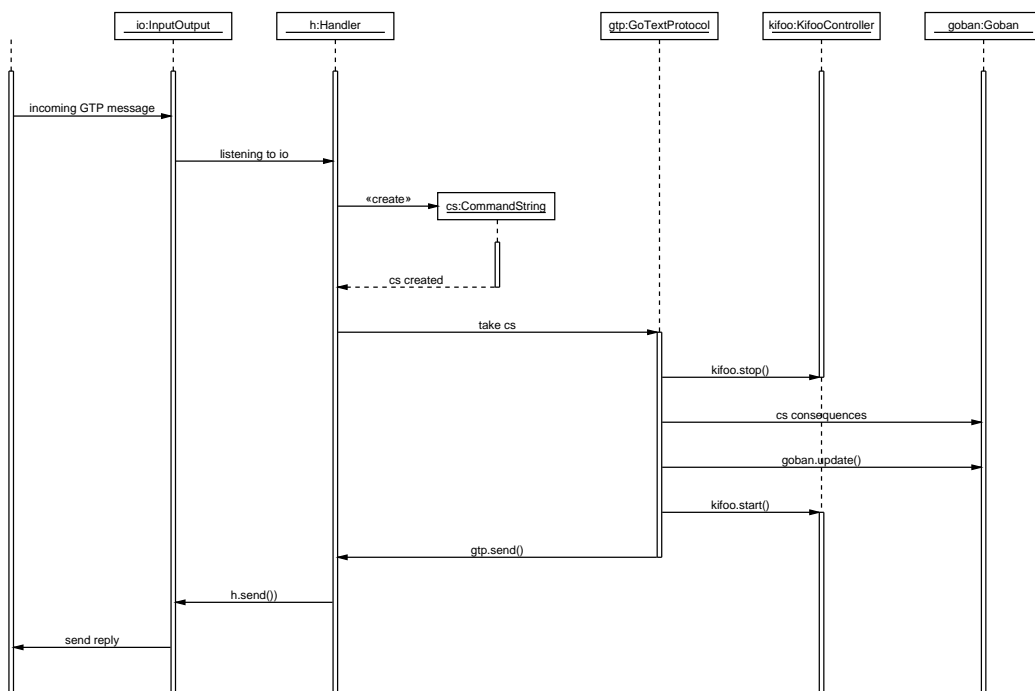


Abbildung 3.1: Kommunikationsdiagramm

sie an `CommandString` weiter. Dort wird die Eingabe verarbeitet, so dass aus der zusammenhängenden Nachricht die Einzelteile entstehen, wie sie in [Abbildung 2.1](#) zu sehen sind. Die aufbereitete Nachricht wird der Implementation von `Parser` bereitgestellt. Die Implementation `Parser` wird im folgenden Protokoll-Klasse genannt. Die Struktur der Protokoll-Klassen ist in [Abbildung 3.2](#) dargestellt.

Die Protokoll-Klassen `GoTextProtocol` und `KeseidoGoServer` übersetzen

diese dann in Befehle für Kifoo und sorgen für deren Ausführung. Kifoo erzeugt als Antwort darauf Rückgabewerte, die versandt werden. Sie werden von den Protokoll-Klassen an den Handler zurückgegeben und über die Netzwerkschnittstelle nach draußen weitergegeben.

Um das Go-Text-Protokoll zu implementieren, ergibt es durchaus Sinn, einen Parser zu verwenden, der in JFlex [JF1] erstellt wird. Es hat sich gezeigt, dass es für weitere Entwicklungen und Entwickler schwerer wird, Neues hinzuzufügen, da Kenntnisse über JFlex vorausgesetzt werden müssen. Daher gibt es eine Klasse `CommandString`, die die Eingabe in einzelne Stücke zerlegt und für die Weiterverarbeitung vorbereitet. Diese Klasse implementiert eine Art von lexikalischer Analyse. Dort wird noch nicht überprüft, ob die Eingabe korrekt ist, sondern nur, ob sie der Form entspricht, die für GTP erwartet wird (siehe Abbildung 2.1). Der nächste Verarbeitungsschritt erfolgt durch die Implementation des Java Interfaces `Parser`. Die Klasse `GoTextProtocol` implementiert die Schnittstelle und sorgt für die zu erfolgenden Aktionen von Kifoo. Die Implementation muss prüfen, ob das darin enthaltene Kommando einem Befehl des GTP zugeordnet werden kann und ob die angegebenen Parameter dem Befehl entsprechen. Ist der Befehl enthalten, so wird er ausgeführt und ein entsprechender Rückgabewert von Kifoo erzeugt. Wenn ein Kommando nicht gefunden wurde, wird ein Rückgabewert erzeugt, dass dieses Kommando unbekannt ist. Passen Parameter nicht zum Kommando, so wird ebenfalls ein Rückgabewert zurückgegeben, in diesem Fall ein Fehler.

## 3.2 Dateiformate

Im Wesentlichen wird in Kifoo nur SGF unterstützt, da die anderen Dateiformate mit Hilfe von Programmen [sgfb] in SGF transferiert werden können. Seitens der bekanntesten und gebräuchlichsten Go-Programme und -Server, gibt es noch keine zufriedenstellende XML-Format-Unterstützung. Es existieren einige konkurrierende XML-Formate einzelner Go-Programme. Einiges spricht dafür, das von den Entwicklern entworfene XML-Format XGF zu verwenden, da SGF selbst zwar sehr bekannt ist, aber leider noch in einer

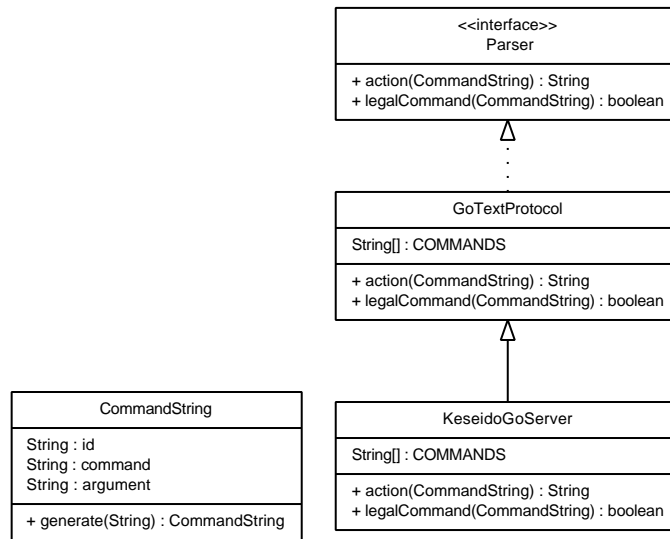


Abbildung 3.2: Protokoll-Klassen

Fassung besteht, die als Diskussionsgrundlage dient und somit als Dateiformat entfällt. Es existieren einige Java-basierte Implementationen von Lesern für SGF-Daten. Deren Quelltext ist zum Teil schwer nachzuvollziehen und schlecht lesbar, so dass in Betracht kommt, für Kifoo eine neue Implementation zu verwenden. JFlex wird in Kifoo nicht verwendet, da der entstehende Quelltext des Parsers nicht gut lesbar ist, wenn man die Arbeitsweise von JFlex nicht kennt.

Im allgemeinen werden in JFlex einzelne Wörter mit regulären Ausdrücken erkannt und entsprechend verarbeitet. Diese regulären Ausdrücke wurden in der Implementation wieder verwendet. Die in der SGF-Datei enthaltenen Textstücke werden mit Hilfe von regulären Ausdrücken erkannt und in GTP-verständliche `CommandStrings` verarbeitet und an das `GoTextProtocol` weitergereicht.

# Kapitel 4

## Tests und Experimente

Es ist sehr mühsam, ein Computer-Go-Programm zu testen, wenn man Hunderte Spiele selbst spielen muss. Dass Erweiterungen zu einem besseren und stärkeren Spiel führen, lässt sich auf diese Weise nur sehr schwer erkennen. Aus diesem Grund haben sich einige Entwickler von Computer-Go-Programmen Testumgebungen geschrieben und Partien zusammengestellt, an denen man die größten Fehler oder auch Verbesserungen schnell erkennen kann. Es handelt sich bei solchen Umgebungen um Spiele oder auch Spielabschnitte, für die der beste Zug bekannt ist, so dass ein Computer-Go-Programm diesen auch spielen können sollte. Es ist natürlich nicht das Ziel, die Tests möglichst gut zu bestehen, wenn man im regulären Spiel sonst gegen Gegner verliert. Die Tests helfen, die Entwicklung eines Computer-Go-Programms zu überwachen und gute Ideen weiterzuentwickeln. Des Weiteren ist es nützlich, Kifoo gegen Spieler auf öffentlichen Servern antreten zu lassen, um das Spiel gegen andere Spieler zu bewerten. Eine Auswertung von 1000 Spielen findet sich im Abschnitt "Testspiele auf KGS Server".

### 4.1 Regressionstestumgebungen

Gnu Go ist eines der bekanntesten Go-Programme. Es bietet Testskripte, die in GTP geschrieben worden sind, die SGF-Daten einlesen lassen, gegen die man seine Computer-Go-Ki antreten lassen kann. Des Weiteren gibt es noch

die Computer Go Test Collection (CGTC) [Mü] , die auch eine GTP sprechende Testumgebung anbietet. Dies bietet eine gute Grundlage zum Lernen von Go für Kifoo. Kifoo spielt je nach Bedenkzeit unterschiedlich, siehe Abbildung 4.1. Um diesem Umstand Rechnung zu tragen, wurde eine Möglichkeit eine Bedenkzeit anzugeben hinzugefügt. Die unterschiedlichen Bedenkzeiten

```
./kifoo-eval.sh kifoo-120s.9x9.tst
...
Summary: 9/62 passes. 5 unexpected passes, 13 unexpected failures
(a) 9x9 Tests mit 120s Bedenkzeit

./kifoo-eval.sh kifoo-240s.9x9.tst
...
Summary: 6/62 passes. 4 unexpected passes, 15 unexpected failures
(b) 9x9 Tests mit 240s Bedenkzeit
```

Abbildung 4.1: Testumgebung

geben Aufschluss, wie schnell oder langsam Kifoo spielt, um zu seinen Ergebnissen zu kommen. Eine Möglichkeit, die Tests zu überprüfen, ist die Ausgabe in `psgo`-Text. Es ist möglich sich die Ausgangssituation mit dem bestmöglichen Zug anzuschauen sowie die Lösung Kifoo. Diese Übersicht ist hilfreich für die Entwicklung von Kifoo. Abbildung 4.2 zeigt eine Testpartie mit Ausgangssituation, Lösung und Kifoo's Spielzug. Die Tests und die Übersicht bieten eine gute Basis zum Evaluieren des Spielverhaltens von Kifoo.

## 4.2 Testspiele auf KGS Server

Testspiele gegen Menschen und Computer-Go-Programme sind hilfreich zum Entdecken von Spielfehlern. Wir haben im Juni 2009 1000 Spiele mit einer Brettgröße von  $9 \times 9$  auf dem KGS Server durchgeführt [kif] , um zu testen, wie stark unser Programm gegen menschliche Spieler bzw. Computer-Go-Programme ist, denen eine Spielstärke zugeordnet ist. Fast die Hälfte der Spiele fanden gegen Gegner statt, denen keine Spielstärke zugeordnet ist. Zur Erläuterung siehe Statistikgrafiken Abbildung 4.3 und 4.4. Die Spielstärken gliedern sich in zwei Bereiche, einmal den Amateur-Bereich von 28 kyu bis 1 kyu und den professionellen Bereich von 1 dan bis zu 6 dan. Die Werte, die

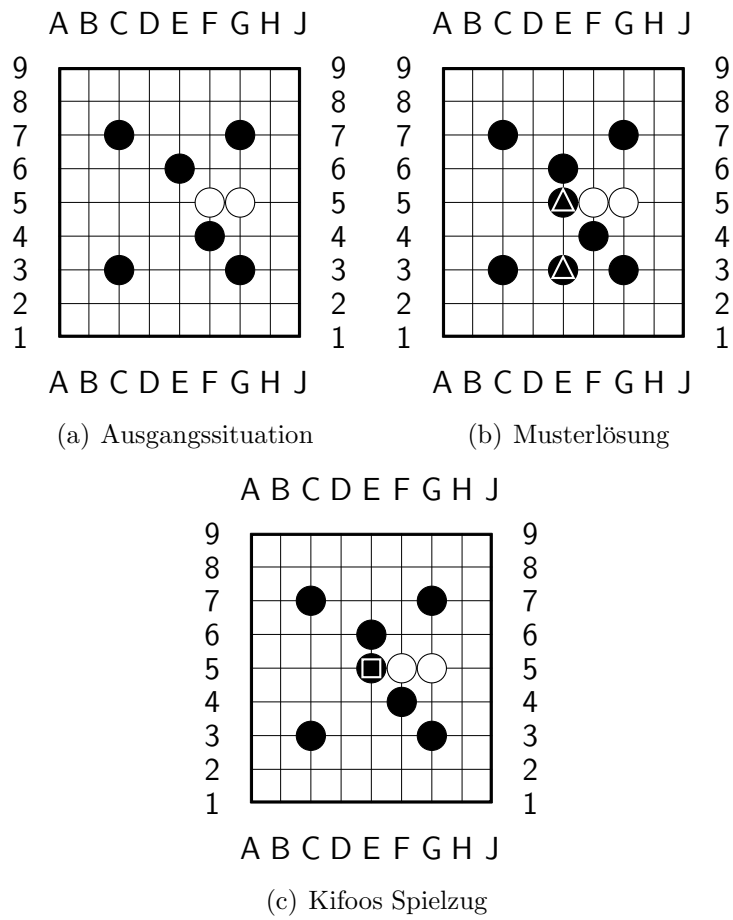


Abbildung 4.2: Beispielausgabe von Kifoos Testumgebung

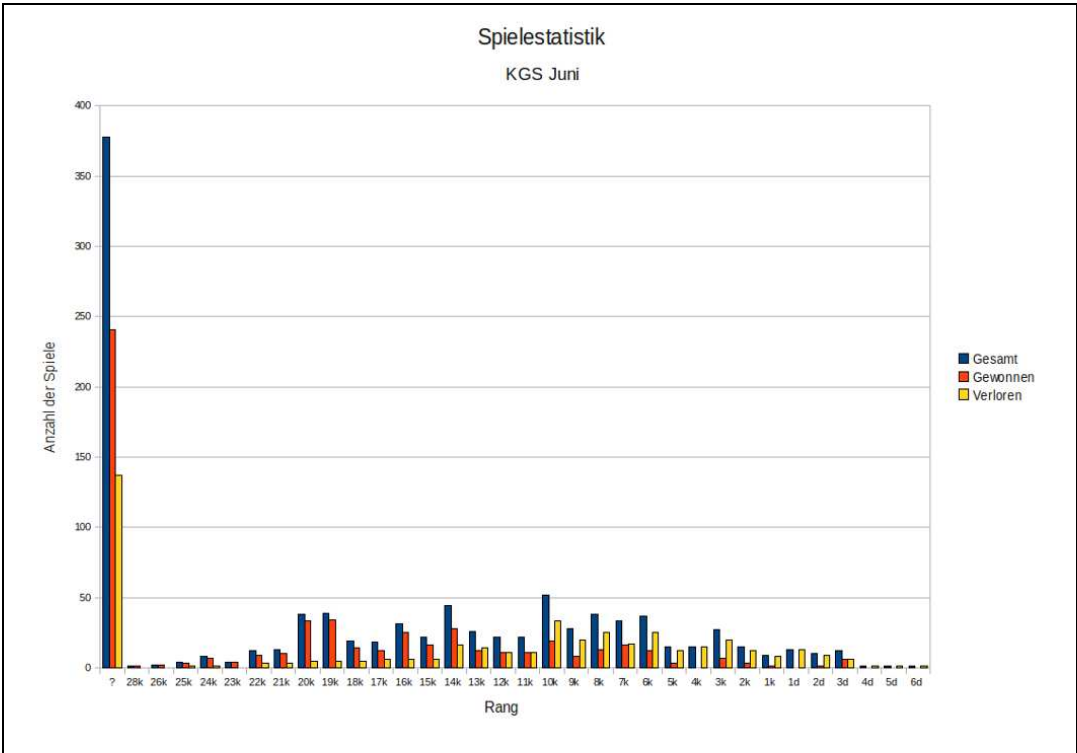


Abbildung 4.3: Gesamtstatistik aller 1000 Spiele

diesen nicht zugeordnet werden konnten, sind im ? Bereich. Zur Auswertung der gewonnenen und verlorenen Spiele wurden die Spiele in Gnu-Go eingegeben und eine Bewertung des Spiels vorgenommen. In der unteren Grafik ist der Bereich zwischen 14 kyu und 10 kyu interessant, weil sich die gewonnenen Spiele gegenüber den verlorenen Spielen in Waage halten.

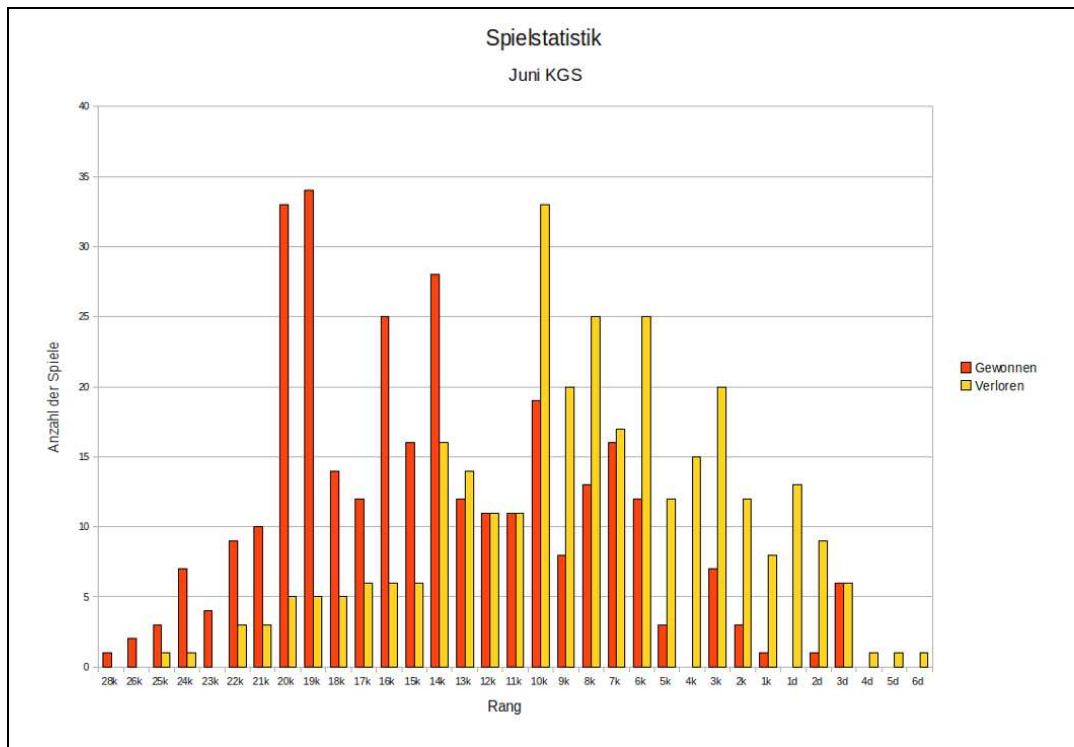


Abbildung 4.4: Statistik der Spiele mit Gegnern von Rang

# Kapitel 5

## Ausblick

Es bleibt noch einiges zu tun. Kifoo bietet SGF, GTP und `psgo` für Ein- und Ausgaben. Testumgebungen können mit Kifoo genutzt werden [Fou] [Mü]. Die Erweiterungen befähigen Kifoo im Internet zu spielen und Spiele zur Dokumentation auszugeben. Kifoo könnte in Zukunft bei Computer-Go-Turnieren, die im Internet stattfinden, teilnehmen [Com]. Kifoo sollte dauerhaft im Computer-Go-Bereich auf dem KGS-Go-Server spielen, um einen Rang zu erhalten, um so die Spielstärke von Kifoo über einen längeren Zeitraum zu bestimmen.

# Abbildungsverzeichnis

1.1	Spielbrettgröße 19 . . . . .	2
1.2	Kifoo Logo . . . . .	4
2.1	Struktur der GTP-Kommandos . . . . .	8
2.2	GTP: allgemeine Kommandos . . . . .	8
2.3	GTP: Regressionskommandos . . . . .	8
2.4	GTP: Turnierkommandos . . . . .	9
2.5	KGS GTP Erweiterungen . . . . .	9
2.6	Beispiel SGF . . . . .	10
2.7	Beispiel ishi . . . . .	12
2.8	sgf2dg Ausgabe des SGF-Beispiels . . . . .	14
2.9	Beispiel Ausgaben . . . . .	15
2.10	Darstellungen in Latex-Quelltext . . . . .	16
3.1	Kommunikationsdiagramm . . . . .	18
3.2	Protokoll-Klassen . . . . .	20
4.1	Testumgebung . . . . .	22
4.2	Beispielausgabe von Kifoo's Testumgebung . . . . .	23
4.3	Gesamtstatistik aller 1000 Spiele . . . . .	24
4.4	Statistik der Spiele mit Gegnern von Rang . . . . .	25

# Literaturverzeichnis

- [Aug] Reid Augustin. `sgf2dg`. Convert Smart Go Format (SGF) files to diagrams similar to those seen in Go books and magazines <http://search.cpan.org/~reid/Games-Go-Sgf2Dg-4.211/sgf2dg>.
- [Bos] V. Bos. `psgo`. PSGO is a LaTeX2e package to draw Go diagrams in LaTeX <http://www.ctan.org/tex-archive/graphics/pstricks/contrib/psgo>.
- [Com] Liste von Computer-Go Turnieren <http://www.computer-go.info/events/future.html>.
- [DGO] Deutscher Go Bund Webseite <http://www.dgob.de>.
- [Far02] Gunnar Farnebäck. Go text protocol, 2002. Spezifikation: <http://www.lysator.liu.se/~gunnar/gtp>.
- [Fot90] David Fotland. Ishi (standardformat) spezifikation, 1990. Standard format Ishi Spezifikation <http://www.britgo.org/tech/ishispec.html>.
- [Fou] Free Software Foundation. Gnu go. GNU Go is a free program that plays the game of Go <http://www.gnu.org/software/gnugo/gnugo.html>.
- [GoG] Gogui. GoGui is a graphical user interface to programs that play the board game Go and support the Go Text Protocol, such as GNU Go. <http://gogui.sourceforge.net>.

- [Groa] René Grothmann. jago xml. Spezifikation <http://jagoclient.sourceforge.net/Documentation/xml.html>.
- [Grob] René Grothmann. Jagoclient. Webseite <http://jagoclient.sourceforge.net>.
- [HMG97] Hiroyuki Iida Hitoshi Matsubara and Reijer Grimbergen. Chess, shogi, go, natural developments in game research. Technical report, Electrotechnical Laboratory, 1-1-4 Umezono, Tsukuba, Ibaraki, Japan 305, 1997.
- [HSU07] FENG-HSIUNG HSU. Cracking go, 2007. <http://spectrum.ieee.org/computing/software/cracking-go>.
- [igs] Pandanet Internet Go Server <http://www.pandanet.co.jp>.
- [JF1] JFlex - The Fast Scanner Generator for Java <http://jflex.de/>.
- [KGSa] Informationen über KGS auf Sensei's Library <http://senseis.xmp.net/?KGS>.
- [KGSb] Kgs go server. Webseite <http://www.gokgs.com>.
- [Kie87] A. Kierulf. Human-computer interaction in the game of go. In *Proceedings of the Second International Symposium on Methodologies for intelligent systems*, pages 481–487, Amsterdam, The Netherlands, The Netherlands, 1987. North-Holland Publishing Co.
- [kif] kifooai. Archiv von Kifoo's Partien Juni 2009 <http://www.gokgs.com/gameArchives.jsp?user=kifooAI&oldAccounts=t&year=2009&month=6>.
- [kif09] Kifoo, 2009. Kifoo-Projekt Webseite <http://dev.spline.de/trac/kifoo>.
- [Lib] Liberty fileformat. Spezifikation <http://gobase.org/software/liberty/index.htm>.

- [Mü] Martin Müller. Computer go test collection. Original Anhang von [Mü95] <http://webdocs.cs.ualberta.ca/~mmueller/cgo/appendix.html>.
- [Mü95] Martin Müller. *Computer Go as a Sum of Local Games: An Application of Combinatorial Game Theory*. PhD thesis, Swiss Federal Institute of Technology Zürich, 1995.
- [Nij06] Emil H.J. Nijhuis. Learning Patterns in the Game of Go. Master's thesis, Universiteit van Amsterdam, December 2006.
- [Ser] KGS Go Server. Kgs gtp klient. Spezifikation <http://www.gokgs.com/download.jsp>.
- [sgfa] Sgf diskussion. <http://www.britgo.org/tech/sgfspec>.
- [sgfb] sgf2misc. sgf2misc, go diagram print and game conversion filter <http://gobase.org/software/sgf2misc>.
- [SGFc] Smart game format. Spezifikation <http://www.red-bean.com/sgf>.
- [sgfd] Smart game format historie. <http://www.red-bean.com/sgf/history.html>.
- [Wed] Nick Wedd. Human-computer go challenges. <http://www.computer-go.info/h-c/index.html>.
- [XGF02] Xgf - an xml game format, 2002. Spezifikation <http://www.red-bean.com/sgf/xml>.