The Syntax of Chess Moves in the LaTeX Package "skak" (Draft)

Dirk Bächle

February 20, 2025

Abstract

This short document contains some thoughts and ideas about the "to be supported" syntax of chess moves, including SAN (Short Algebraic Notation) as well as LAN (Long Algebraic Notation).

The main purpose of this draft is to specify a concrete set of allowed chess moves. In the next step, extensive test routines for this syntax should be generated that can be used to verify the final implementation...

Stage 1: Parsing "`\mainline`"/"`\variation`"

The commands "`\mainline`" and "`\variation`" should accept a nonemtpy list of space separated "move tokens" (MT):

`\mainline`{*MT_list*}   or   `\variation`{*MT_list*}

where

$$\begin{aligned} MT\_list &\rightarrow MT\ \textbf{space}\ MT\_list \\ &\mid MT \end{aligned}$$

using BNF (Backus-Naur Form) notation with '**space**' as the terminal symbol for the character ' ' (ASCII code 32_{10}).

Stage 2: Splitting off move numbers

Each MT may be either a chess move (CM), a move number (MN) or a token like "`2.Kg1`" that combines both:

$$\begin{aligned} MT &\rightarrow MN\ CM \\ &\mid CM \\ &\mid MN \end{aligned}$$

The only task of this stage is to separate "combined tokens" (first rule in the production above) and, therefore, supply the next stage 3 with a steady stream of single move tokens (M).

> Remark: This “separation” probably has to be done by inspecting the whole token character for character. Thus, it appears to be efficient to already collect information about the move number MN (White/Black to move, getting the number itself, suppressing leading zeros) in this early stage...

No semantic checking is done here, i.e. this stage doesn't detect errors like “`2. Kg1 3. Bf4`”.

Stage 3: Parsing single move tokens

At this point, we can be sure to get either a chess move CM or a move number MN:

$$\begin{aligned} M &\rightarrow CM \\ &\quad \mid MN \end{aligned}$$

Both can be distinguished well by inspecting the first character of the token, since only move numbers MN may start with a digit (D).
This stage would be the right place to check for the correct order of chess moves and move numbers, i.e. if a MN is encountered the right side (White or Black) should be to move.

Parsing move numbers

Move numbers MN consist of an integer value (N), immediately followed by one or three dots as terminal symbols, signalling whether it's Whites or Blacks move.

$$\begin{aligned} MN &\rightarrow N\ \mathbf{...} \\ &\quad \mid N\ \mathbf{.} \end{aligned}$$

An integer value N is a nonempty list of single digits (D)...

$$\begin{aligned} N &\rightarrow D\ N \\ &\quad \mid D \end{aligned}$$

... where each digit D should be contained in the set of terminal symbols 0–9.

$$D \rightarrow \mathbf{0}|\mathbf{1}|\mathbf{2}|\mathbf{3}|\mathbf{4}|\mathbf{5}|\mathbf{6}|\mathbf{7}|\mathbf{8}|\mathbf{9}$$

Here, the move number can be checked against the internal move counter if necessary.

Parsing chess moves

A chess move CM starts with the “move specification” (MS), giving information about the piece that is to move and the source and destination squares. Additionally, an arbitrary number of character tokens may follow as a move comment (MC). So, after enough “hints” for executing the move are collected, i.e. MS was “matched”, the rest of the token is regarded as comment MC and is output unchanged.

$$\begin{aligned} CM &\rightarrow MS\ MC \\ &\mid MS \end{aligned}$$

Before defining the move specification MS itself a few helping nonterminal symbols are introduced. Please, remember that bold characters within the "rules" denote terminal symbols…

1. A "piece" character (P)

   $$P \rightarrow \mathbf{K}|\mathbf{Q}|\mathbf{B}|\mathbf{N}|\mathbf{R}$$

   Remark: This definition uses the english letters for the single pieces. The real implementation should be independent of the used language.

2. A "file" character (f)

   $$f \rightarrow \mathbf{a}|\mathbf{b}|\mathbf{c}|\mathbf{d}|\mathbf{e}|\mathbf{f}|\mathbf{g}|\mathbf{h}$$

3. A "rank" character (r)

   $$r \rightarrow \mathbf{1}|\mathbf{2}|\mathbf{3}|\mathbf{4}|\mathbf{5}|\mathbf{6}|\mathbf{7}|\mathbf{8}$$

4. A "capture" character ($\bowtie$)

   $$\begin{aligned} \bowtie &\rightarrow \mathbf{-} \\ &\mid \mathbf{x} \\ &\mid \epsilon \end{aligned}$$

   where 'ϵ' denotes the "empty symbol".

Move specifications (MS) can be subdivided into the following groups:

- Starting with a "piece" character
  - With source square

    $$\begin{aligned} MS &\rightarrow P\ f\ r \bowtie f\ r \\ &\mid P\ f \bowtie f\ r \\ &\mid P\ r \bowtie f\ r \end{aligned}$$

    Examples: "`Qf3-f4`", "`Ree6`", "`N1xc3`"
  - Only a destination square present

    $$MS \rightarrow P \bowtie f\ r$$

    Example: "`Nxc4`"
- Without leading "piece" character
  - With source square

$$
\begin{aligned}
MS \rightarrow\ & f\ r \bowtie f\ r\ P \\
\mid\ & f \bowtie f\ r\ P \\
\mid\ & f\ r \bowtie f\ r \\
\mid\ & f \bowtie f\ r
\end{aligned}
$$

Examples: “`f7-f8R`”, “`dxe8B`”, “`g2-g4`”, “`fxe6`”

- Only a destination square present

$$
\begin{aligned}
MS \rightarrow\ & f\ r\ P \\
\mid\ & f\ r
\end{aligned}
$$

Examples: “`e8R`”, “`a6`”

- Castlings

$$
\begin{aligned}
MS \rightarrow\ & \textbf{O-O-O} \\
\mid\ & \textbf{O-O}
\end{aligned}
$$

Both of these terminals use the letter ’O’ (ASCII code 79_{10}) and not the digit ’0’ (ASCII code 48_{10})!

Final remark

This text tries to provide only the syntax for the frontend of the new “move machine”. All the things that happen after the input was “matched” and information was drawn out of the given “chess moves” to the largest extent, i.e. the semantic actions, are beyond the scope of this document.
So, at the moment it is perfectly legal and syntactically correct to say “`4. gxf8K`”...