PDN 3.0 Standard

Release 3.0

Wieger Wesselink

Jan 09, 2025

CONTENTS

CHAPTER

ONE

INTRODUCTION

This document defines the official portable draughts notation standard, PDN 3.0. The goal of this document is to give a formal definition of PDN. Nowadays there are many conflicts between draughts programs regarding their interpretation of valid PDN. This document aims to take away the sources of conflicts, by defining grammars, and by giving additional restrictions. Note that the PDN definition as given in this document slightly deviates from earlier versions. This has been done to make parsing PDN easier. Furthermore, some extensions are defined to support setting up positions and to add clock times.

In [Wikipedia], [Sage], [Nemesis] and [Grimminck] earlier versions of PDN definitions can be found. PDN was derived from the PGN standard in chess, see [PGN]. The PDN proposal in this document takes the PDN 2.0 draft [Nemesis] as a starting point, since it is more complete and precise than the earlier ones. Several additions, restrictions and extensions to the PDN 2.0 draft are made.

This document first discusses PDN tags, then a grammar for PDN 3.0 is given, followed by some PDN extensions, and some example implementations of the grammar. The design of the grammar was influenced by a number of issues of the existing PDN standards. These issues are discussed in the *PDN parsing issues* section.

For convenience a Java Webstart PDN 3.0 Checker is available, that can be used for checking the validity of a PDN file.

1.1 PDN Example

A PDN file consists of a section of tags (key-value pairs between brackets) followed by a section with moves and variations. A typical example is:

```
[Event "FMJD World Championship"]
[Site "Hardenberg, NED"]
[Date "2007.05.19"]
[Round "7"]
[White "Mikhalchenko,I."]
[Black "Ndjofang,J."]
[Result "0-2"]
[GameType "20"]
[WhiteTime "1:36"]
[BlackTime "1:17"]

 1.32-28 17-22  2.28x17 11x22  3.37-32  6-11  4.41-37 12-17  5.46-41  8-12
 6.34-30  2-8   7.30-25 19-23  8.35-30  1-6   9.40-35 13-19 10.31-27 22x31
11.36x27  9-13 12.33-28  4-9  13.41-36 17-22 14.28x17 11x31 15.37x26 23-28
16.32x23 19x28 17.42-37 20-24 18.30x19 14x23 19.37-31 16-21 20.26x17 12x21
21.31-27 21x32 22.38x27  6-11 23.47-42 15-20 24.25x14 10x19 25.39-33 28x39
26.44x33  8-12 27.42-38 23-28 28.33x22 12-17 29.49-44 17x28 30.38-33 28x39
31.44x33 18-22 32.27x18 13x22 33.43-38 19-23 34.38-32 11-17 35.32-27 22x31
36.36x27  9-13 37.45-40 13-18 *
```

1.2 Reading and writing

Programmers are encouraged to follow the PDN 3.0 standard closely when writing PDN. When reading, it is recommended to treat the input much more liberally. To this end an example of a liberal reading grammar is given. Also the character encoding requirement can be relaxed.

1.3 Acknowledgements

Thanks to everyone who took part in the discussions on the World Draughts Forum [Forum], and to Rein Halbersma for leading the review.

1.4 References

CHAPTER

TWO

CHARACTER ENCODING

For PDN 3.0 documents the UTF-8 character encoding is required. Note that this includes ASCII encoded documents. When reading PDN, it is recommended to accept other encodings too, like ISO 8859/1 (Latin 1).

Note that PDN differs from PGN in this respect, see also http://www.saremba.de/chessgml/standards/pgn/pgn-complete.htm#c4.1. The reason for this choice is that UTF-8 supports many different character sets like Cyrillic, Chinese, Japanese etc. UTF-8 is also used in the XHTML standard.

There is no maximum line length defined for a PDN document. So it is allowed to put an entire game on one line.

CHAPTER

THREE

PDN TAGS

This section gives an overview of predefined PDN tags. Tags are key-value pairs between brackets, in which the value is surrounded by double quotes. More extensive documentation about most of the tags given here can be found in [PGN]. The number of predefined tags has been kept limited. It is allowed to add user defined tags. Note that a key must start with a capital.

We adopt the rule that a tag is omitted if it has no value. This deviates from [PGN], that defines an explicit *no value* for some tags in the form of a minus sign `"-"`. To denote that a tag has an unknown value, in most cases a `"?"` value can be used. For example `[Event "?"]` denotes that the event of the game was unknown. Some tags have a more specific unknown value. Instead of a `"?"` value, it is also allowed to use the empty string `""`.

This section gives an overview of the predefined tags, followed by some examples and additional details about the tags.

3.1 'Mandatory' tags

Tag	Description	Unknown value
Event	Name of the tournament or match event, see [Nemesis]	"?"
Site	Location of the event, see [Nemesis]	"?"
Date	Starting date of the game, see [Nemesis]	"????.??.??"
Round	Playing round ordinal of the game, see [Nemesis]	"?"
White	Player of the White pieces, see [Nemesis]	"?"
Black	Player of the Black pieces, see [Nemesis]	"?"
Result	Result of the game, see [Nemesis]	"*"

N.B. The name *mandatory tags* is misleading, since these tags are not mandatory for an arbitrary PDN file. It is however recommended to include all of the mandatory tags in tournament games.

3.2 Result tag

The `Result` tag is used to specify whether a game ended in a win, draw or a loss. Each game type has a specific set of allowed values for the `Result` tag.

ResultType	Allowed result values
Default	1-0, 0-1, 1/2-1/2, 0-0, *
International	2-0, 0-2, 1-1, 0-0, *

In the section *GameType tag* it is specified which result values belong to which game type.

The value * denotes that the game was unfinished, or there is no result available. The value `0-0` denotes that the game was declared lost for both players.

When reading a PDN document, it is recommended to accept arbitrary strings as results. Note that when the `ResultFormat` tag is set, the set of allowed values for the `Result` tag is overruled.

Examples:

```
[Result "1/2-1/2"]
```

Sometimes tournaments are being played with different result values. To this end the `ResultFormat` tag is defined. Below a table is given for some common result formats:

ResultFormat	Allowed result values
Plus draw	2-0, 0-2, 1-1, 0-0, 1+ - 1-, 1- - 1+, *
Delfts	2-0, 0-2, 1 1/2 - 1/2, 1/2 - 1 1/2, 1-1, 0-0, *
Goes	2-0, 0-2, …, -0.98-1.02, -0.99-1.01, 1-1, 1.01-0.99, 1.02-0.98, …, 0-0, *

When the `ResultFormat` tag is set, the `Result` tag must have a corresponding allowed value.

Examples:

```
[ResultFormat "Plus draw"]
[Result "1+ - 1-"]
```

3.3 Player related tags

Tag	Description
WhiteTitle, BlackTitle	FMJD titles of the players
WhiteRating, BlackRating	FMJD rating
WhiteNA, BlackNA	E-mail or network addresses of the players
WhiteType, BlackType	Player types (“human” or “program”)

The tags `WhiteRating` and `BlackRating` are named WhiteElo and BlackElo in chess.

3.4 Event related tags

Tag	Description
EventDate	Starting date of the event
EventSponsor	Sponsor of the event
Section	Playing section of a tournament (e.g., “Open” or “Reserve”)
Stage	Stage of a multistage event (e.g., “Preliminary” or “Semifinal”)
Board	Board number in a team event or in a simultaneous exhibition

3.5 Game related tags

Tag	Description
GameType	Type of the game, see *GameType tag*
FEN	The position at the start of the game, see *FEN tag*
PlyCount	The number of ply (moves) in the game
Termination	Describes the reason for conclusion of the game, see [PGN]

The `GameType` tag is specific for draughts, and is used to distinguish between the different draughts variants.

3.6 Clock related tags

Tag	Description
TimeControl	Time control settings for both players, see [PGN]
TimeControlWhite	Time control settings for the white player
TimeControlBlack	Time control settings for the black player
WhiteTime	Time used by the White player at the end of the game
BlackTime	Time used by the Black player at the end of the game

The `WhiteTime` and `BlackTime` tags are new. It is common practice to record the time used by both players, so it seems useful to define a tag for it. The `TimeControlWhite` and `TimeControlBlack` tags can be used when the players start with different times on the clock. This is for example the case in Georgiev-Lehmann tie-breaks.

3.7 Time and date tags

Tag	Description
Time	Time-of-day value in "HH:MM:SS" format
UTCTime	Time-of-day in Universal Coordinated Time format
UTCDate	Date in Universal Coordinated Time format

3.8 Miscellaneous tags

Tag	Description
Annotator	Identifies the annotator or annotators of the game

3.9 Problemism related tags

Tag	Description
Author	Author(s) of the analysis or composition
Publication	Original publication of the analysis or composition
PublicationDate	Date of the original publication

These tags are new. PDN can be used to store databases with problems, so it seems useful to define tags to support this.

3.9.1 Details and Examples

Event Tag

The Event tag specifies the event. Abbreviations are to be avoided.

Example:

```
[Event "FMJD World Championship"]
```

Site Tag:

The Site tag specifies the location. Use IOC country codes to denote countries.

Examples:

```
[Site "New York City, NY USA"]
[Site "St. Petersburg RUS"]
[Site "Riga LAT"]
```

Date Tag

The Date tag must be specified in `YYYY.MM.DD` format. Question marks may be used for unknown fields.

A regular expression for Date values is:

```
([0-9]{4}|[?]{4})\.([0-9]{2}|[?]{2})\.([0-9]{2}|[?]{2})
```

Examples:

```
[Date "1996.12.28"]
[Date "2007.??.??"]
```

Round Tag

Examples:

```
[Round "1"]
[Round "3.1"]
[Round "4.1.2"]
```

White/Black Tag

The White and Black tag are used to specify the names of the players. The family or last name appears first.

Examples:

```
[White "Wiersma, Harm"]
[White "van der Wal, Jannes"]
[White "Dragon v.4.0"]
[White "Schwarzman, A."]
```

WhiteTime/BlackTime Tag

The WhiteTime and BlackTime tags specify the amount of time that the players have used during the game. Note that these tags do not exist in earlier versions of the PDN standard. It is common practice to record these times, hence it seems logical to define a tag for it.

Clock times are specified in `[H]H:MM[:SS]` format. Note that in practice also `[H]H.MM[.SS]` is used.

When Fischer time controls are used it makes more sense to record the remaining time on the clock. A notation is needed to specify this.

Examples:

```
[WhiteTime "1:59:20"]
[BlackTime "1:17:28"]
```

TimeControl Tag

The time controls are specified using the TimeControl tag.

Time control values should match with the following grammar:

```
// Productions
TimeControl   : UNKNOWN | NOTIME | CompositeTime
TimeElement   : MOVES_SECONDS | INCREMENTAL | SUDDENDEATH | SANDCLOCK
CompositeTime : TimeElement (COLON TimeElement)*

// Tokens
MOVES_SECONDS : "[0-9]+\/[0-9]+"
INCREMENTAL   : "[0-9]+\+[0-9]+"
SUDDENDEATH   : "[0-9]+"
SANDCLOCK     : "\*[0-9]+"
UNKNOWN       : "\?"
NOTIME        : "\-"
COLON         : "\:"
```

Examples:

```
[TimeControl "40/7200:3600"]          { 40 moves in 7200 seconds, 3600 seconds␣
↪for the rest of the game }
[TimeControl "4800+60"]               { 80 minutes with increment of 60␣
↪seconds/move }
[TimeControl "40/7200:3600+60"]       { 40 moves in 2 hours, 1 hour for the␣
↪rest of the game with increment of 60 seconds/move }
[TimeControl "40/7200:20/2400:600+5"] { 40 moves in 2 hours, 20 moves in 40␣
↪minutes, 10 minutes for the rest of the game with increment of 5 seconds/
↪move }
[TimeControl "*120"]                  { 2 minutes for a "sandclock" or
↪"hourglass" control period, more suitable usage with physical sandclock }
```

CHAPTER

FOUR

PDN GRAMMARS

This section defines some PDN grammars. First a PDN 3.0 grammar is given, with some additional restrictions. The goal of these restrictions is to make PDN easier to parse, and thus to make it easier for programmers to support PDN 3.0. Also some explanations are given. Finally, a much more liberal *reading* grammar is given that can be used when reading existing PDN files. The grammars that are given in this section can not be parsed using a simple LL(1) parser. See the *PDN Implementation* section for some examples of LL(1) grammars.

4.1 PDN 3.0 Grammar

```
// Game independent productions
PdnFile          : Game (GameSeparator Game)* GameSeparator?
GameSeparator    : ASTERISK
Game             : (GameHeader GameBody?) | GameBody
GameHeader       : PdnTag+
GameBody         : (GameMove | Variation | COMMENT | SETUP | NAG)+
PdnTag           : LBRACKET IDENTIFIER STRING RBRACKET
GameMove         : MOVENUMBER? Move MOVESTRENGTH?
Variation        : LPAREN GameBody RPAREN

// Game dependent productions
Move             : NormalMove | CaptureMove
NormalMove       : Square MOVESEPARATOR Square
CaptureMove      : Square (CAPTURESEPARATOR Square)+
Square           : ALPHASQUARE | NUMSQUARE

// Tokens
MOVENUMBER       : "[0-9]+\.(\.\.)?"
MOVESEPARATOR    : "-"
CAPTURESEPARATOR : "x"
ALPHASQUARE      : "[a-h][1-8]"
NUMSQUARE        : "[1-9][0-9]?"
MOVESTRENGTH     : "([\!\?]+)|(\([\!\?]+\))"
NAG              : "\$[0-9]+"
LPAREN           : "\("
RPAREN           : "\)"
LBRACKET         : "\["
RBRACKET         : "\]"
ASTERISK         : "\*"
SETUP            : "\/[^\/]*\/"
STRING           : "\"([^\"]|\\\")*\""
```

(continues on next page)

(continued from previous page)

```
COMMENT          : "\{[^}]*\}"
IDENTIFIER       : "[A-Z][a-zA-Z0-9_]*"
```

Besides the usual white space characters (spaces, tabs and line endings), also line comments starting with a `%` are allowed. For example

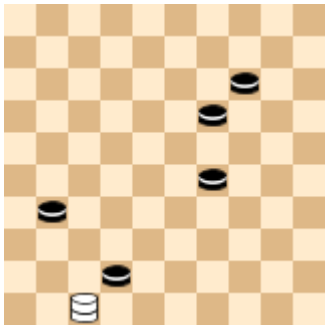
```
% Board game: International Draughts 10x10
[Date "2012.02.01"]
1. 32-28 19-23
```

4.2 PDN 3.0 Restrictions

When writing PDN, the following restrictions should be applied:

1. Spaces are not allowed in the notation of a move. For example, `1 - 7` is not allowed.
2. Spaces are not allowed between a move notation and it's move strength indicator. For example, `32-28 !` is not allowed.
3. The symbol `*` is not allowed as a move strength indicator. Use the `$7` numeric annotation glyph instead. See the *PDN parsing issues* section for an explanation.
4. Only the symbol `*` is allowed as a game separator.
5. Squares of moves may not have leading zeroes. For example `01-07` is not allowed.
6. Moves must be written in the format (Alpha numeric, Numeric or SAN) as it is specified in the Notation attribute of the `GameType` tag. See also section *GameType tag*.
7. Capture moves must be written using the capture separator corresponding to the `GameType`, as it is specified in the table in section *GameType tag*.
8. Ambiguous moves must be written in long notation, i.e. they must contain the full capture sequence. All other moves should use regular notation, i.e. only a begin and an end square.

   Explanation: sometimes the regular notation of a move is ambiguous. For example in the position below the notation `47x36` does not specify exactly which black pieces were captured.

   To resolve this, `47x38x24x13x36` or `47x38x20x9x36` must be chosen.
9. Disambiguated capture sequences have to specify a complete sequence of intermediate squares along the path of the capture. If there is a change in direction, an intermediate square is the square where a turn in direction was made. If there was not a change in direction, the intermediate square is the square immediately behind a captured piece. There is no intermediate square behind the last captured piece, but otherwise leaving out an intermediate square that is not necessary for the disambiguation is forbidden.

For example, in the above diagram `47x24x36` is not allowed, even though it uniquely determines the move. Also `47x33x24x13x36` is not allowed, since 33 is not immediately behind a captured piece.

N.B. The first five restrictions are enforced by the grammar. Other restrictions have to be checked after parsing.

4.3 Explanation

The grammar is given in EBNF format. In the productions the symbol ? stands for 0 or 1 repetitions, * stands for 0 or more repetitions, and + stands for 1 or more repetitions. Tokens are given between double quotes and should be interpreted as regular expressions.

- Strings can have embedded double quotes ", by using the escape sequence \". For example `"An embedded \" quote!"`.
- Comments are placed between braces. For example `{ Start of the game } 33-28 18-22 39-33? { This is a classical mistake }`.
- In existing PDN files games are usually terminated with a result. It can be one of the chess results `1-0, 1/2-1/2, 0-1`, one of the results of international draughts `2-0, 1-1, 0-2`, or a double forfeit `0-0`. Finally the `*` can be used as a terminator.
- A game can not be empty.
- Both numeric moves `32-28` and alpha-numeric moves `a3-b4` are allowed.
- In alpha-numeric moves the separator may be omitted, so `a3b4` is allowed.
- A move number is a number followed by either one dot or three dots, for example `1. 32-28` or `23... 20-25`. The three dots denotes that it is a black move.
- Moves can be annotated using a move strength indicator right after their notation, for example `10-15!` or `29-23(?)`.
- Moves can also be annotated using numeric annotation glyphs (NAGs). For example `$1` has the same meaning as the move strength indicator `!`.
- Comments, variations and NAGs may appear anywhere in the game, in any order. This is less restrictive than in [Nemesis],
- Variations are placed between parentheses. They can be nested arbitrarily, for example: `32-28 19-23 (18-23 38-32 (37-32? 23-29! { Black wins }) 12-18) 28x19 14x23`.
- A setup of a new position is a FEN command surrounded by forward slashes (/), and it can appear anywhere in the game. An example is `/FEN "B:W18,24,27,28,K10,K15:B12,16,20,K22,K25,K29"/`. It is not backward compatible with existing PDN, see the *PDN Extensions* section for an explanation.

4.4 PDN Reading Grammar

```
// Game independent productions
PdnFile          : Game (GameSeparator Game)* GameSeparator?
GameSeparator    : ASTERISK | Result
Game             : (GameHeader GameBody?) | GameBody
GameHeader       : PdnTag+
GameBody         : (GameMove | Variation | COMMENT | SETUP | NAG)+
PdnTag           : LBRACKET IDENTIFIER STRING RBRACKET
GameMove         : MOVENUMBER? Move MOVESTRENGTH?
Variation        : LPAREN GameBody RPAREN
```

(continues on next page)

(continued from previous page)

```
// Game dependent productions
Move              : NormalMove | CaptureMove | ELLIPSES
NormalMove        : Square MOVESEPARATOR Square
CaptureMove       : Square (CAPTURESEPARATOR Square)+
Square            : ALPHASQUARE | NUMSQUARE
Result            : Result1 | Result2 | DOUBLEFORFEIT
Result1           : WIN1 | DRAW1 | LOSS1
Result2           : WIN2 | DRAW2 | LOSS2

// Tokens
WIN1              : "1-0"
DRAW1             : "1\/2-1\/2"
LOSS1             : "0-1"
WIN2              : "2-0"
DRAW2             : "1-1"
LOSS2             : "0-2"
DOUBLEFORFEIT     : "0-0"
ELLIPSES          : "\.\.\."
MOVENUMBER        : "[0-9]+\.(\.\.)?"
MOVESEPARATOR     : "-"
CAPTURESEPARATOR  : "[x:]"
ALPHASQUARE       : "[a-h][1-8]"
NUMSQUARE         : "([1-9][0-9]?)|(0[1-9])"
MOVESTRENGTH      : "([\!\?]+)|(\([\!\?]+\))"
NAG               : "\$[0-9]+"
LPAREN            : "\("
RPAREN            : "\)"
LBRACKET          : "\["
RBRACKET          : "\]"
ASTERISK          : "\*"
SETUP             : "\/[^\/]*\/"
STRING            : "\"([^\"]|\\\")*\""
COMMENT           : "\{[^}]*\}"
IDENTIFIER        : "[A-Z][a-zA-Z0-9_]*"
```

When reading PDN, one must take into account that captures can can contain the full capture sequence, also in the case of non-ambiguous moves. This is for backward compatibility.

Note that the above *reading* grammar does not take everything into account. Some additional constructs that are encountered in practice are:

- Specify unknown moves using a minus sign: `41. - -`.
- Write a move number without a move to specify an empty game: `1. 2-0`.
- Omit move separators in numeric moves: `3228 1823`.

The reading grammar can be extended to handle such cases, but this goes beyond the scope of this document.

In several PDN examples three dots are being used for an unspecified or unknown move. For example `1. ... 7-12` is sometimes used instead of `1... 7-12`. This case has been added to the reading grammar by means of the `ELLIPSES` token.

CHAPTER

FIVE

PDN EXTENSIONS

5.1 Embedded commands

Comments in PDN files may have embedded commands. Embedded commands may appear anywhere inside comments, and they have the following syntax: `[%COMMAND VALUE]`, where `COMMAND` identifies the command and `VALUE` is command-specific syntax.

5.1.1 clk, mct, egt and emt command

The following commands are taken from [DGT]:

Command	Description
clk	Time displayed on a clock (remaining time)
mct	Time displayed on a clock (elapsed time)
egt	Elapsed game time
emt	Elapsed move time

The values of the `clk`, `egt`, `emt` and `mct` commands are in `h:mm:ss` format. Usually `clk` values come from a digital clock, while `mct` values come from a mechanical clock.

Examples

```
1. 31-26 {[%clk 1:55:21]}
```

5.1.2 clock command

The `clock` command is an extension of the `clk` command. An embedded `clk` command should match with the following regular expression:

```
\[%clock\s*(([wWbB])(\d{1,2}:\d\d:\d\d)\s*)(([wWbB])(\d{1,2}:\d\d:\d\d)\s*)?\]
```

Examples

```
1. 31-26 {[%clk w0:00:10 B0:00:03]}

23.44-39 {[%clk 1:05:23]} 18-23 {Optional leading comments
[%clock 0:49:11] optional trailing comments}
```

In cases like this the clock time is connected to the preceding move. It is the preferred way of specifying clock times during live recordings with electronic boards. A clock command may contain one or two clock times. Each of them

may be preceded by a `w` or a `b` to denote the clock time for white or black. If uppercase is used, it means that the clock for this player is running.

5.2 Setup commands

The PDN grammars presented in this document contain an extension for doing setups of a position anywhere in the game. The motivation for having this command is threefold:

- It gives a well-defined way to handle illegal moves, that happen occasionally in tournament practice.
- They can be used to handle move recognition failures of electronic board software.
- In game analysis it is common to make side steps to different positions, for example to a similar position that has occurred before. The setup command allows to incorporate these side steps as normal variations starting with a setup.

A setup command is a FEN setup command surrounded by forward slashes. Note that this extension is not backward compatible with older versions of the PDN standard. This is an intentional choice, since the moves which appear after a setup are ill-defined if the setup is ignored. Setups are changes on the board, and so they should be on the same level as moves. It is therefore not a good idea to model them as embedded commands inside comments.

Null moves

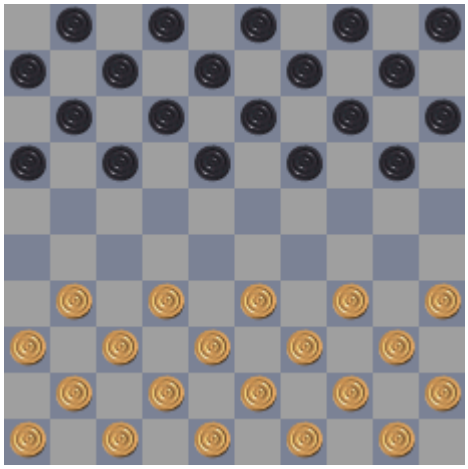
For programmers, null moves are sometimes useful to denote an empty move. A possible notation for a null move, proposed by Gérard Taille, is

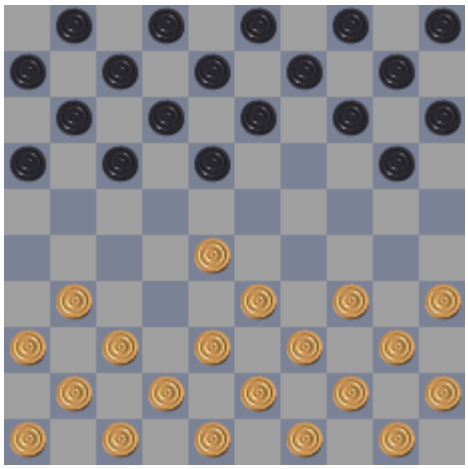
```
/FEN "B::"/
```

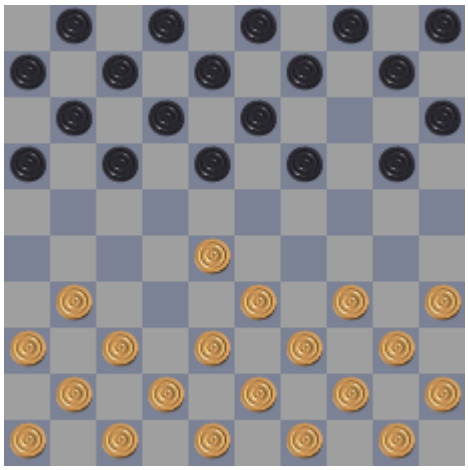
Examples

```
1.31-26 17-21 /FEN "B:B1-16,18-21:W26,28,33-50"/
{ White forgot to make a capture and played 32-28 instead }
```

Suppose a game starts with an illegal move, resulting in the following sequence of positions:







This can be encoded using

```
/FEN "W:W31-50:B1-20"/
/FEN "B:W28,31,33-50:B1-18,20"/
01... 14-19
```

The setup of the initial position is required here. If it was omitted, the second setup would be taken as initial position of the game.

CHAPTER

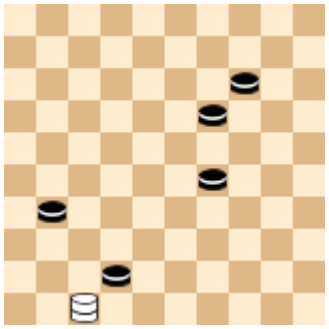
SIX

PDN 3.1 PROPOSALS

This section describes proposals for the next version of the PDN standard.

6.1 Alternative move disambiguation

The current way of move disambiguation (see section *PDN Grammars*) was chosen for backward compatibility. The following alternative is proposed. An ambiguous move is written as usual, followed by the sequence of captured squares between angular brackets < and >. For example, in the following position



the two possible captures are written as `47x36 <42, 29, 14, 31>` and `47x36 <42, 29, 19, 31>`. The motivation for this is that it is less complicated to implement, and easy to understand for humans.

CHAPTER

SEVEN

PDN IMPLEMENTATION

This section contains concrete examples of PDN grammars.

7.1 DParser

[DParser] is a C parser generator.

- `pdn_reading.g` A fairly liberal reading grammar.
- `pdn_writing.g` A PDN 3.0 writing grammar.
- `fen.g` A grammar for FEN strings.
- `timecontrol.g` A grammar for time controls.

7.2 Grammatica

[Grammatica] is a java parser generator.

- `pdn_reading.grammar` A fairly liberal reading grammar.
- `pdn_writing.grammar` A PDN 3.0 writing grammar.
- `fen.grammar` A grammar for FEN strings.
- `timeControl.grammar` A grammar for time controls.
- The Grammatica grammars are LL(1) grammars. They define a move as a token to make this possible.
- The Grammatica grammars contain a workaround for move strengths, since the regular expressions in Grammatica do not behave correctly.

7.3 Toy Parser Generator

[TPG] is a python parser generator.

- `pdn_reading_tpg.py` A fairly liberal reading grammar.
- `pdn_writing_tpg.py` A PDN 3.0 writing grammar.

7.4 Test files

- `games.zip` A collection of PDN games used for testing the grammars.

CHAPTER

EIGHT

PDN EXAMPLES

All of the examples below have been checked with the PDN 3.0 Checker.

An example of an international draughts game in PDN 3.0 format:

```
[Event "FMJD World Championship"]
[Site "Hardenberg, NED"]
[Date "2007.05.19"]
[Round "7"]
[White "Mikhalchenko,I."]
[Black "Ndjofang,J."]
[Result "0-2"]
[GameType "20"]
[WhiteTime "1:36"]
[BlackTime "1:17"]

 1.32-28 17-22  2.28x17 11x22  3.37-32  6-11  4.41-37 12-17  5.46-41  8-12
 6.34-30  2-8   7.30-25 19-23  8.35-30  1-6   9.40-35 13-19 10.31-27 22x31
11.36x27  9-13 12.33-28  4-9  13.41-36 17-22 14.28x17 11x31 15.37x26 23-28
16.32x23 19x28 17.42-37 20-24 18.30x19 14x23 19.37-31 16-21 20.26x17 12x21
21.31-27 21x32 22.38x27  6-11 23.47-42 15-20 24.25x14 10x19 25.39-33 28x39
26.44x33  8-12 27.42-38 23-28 28.33x22 12-17 29.49-44 17x28 30.38-33 28x39
31.44x33 18-22 32.27x18 13x22 33.43-38 19-23 34.38-32 11-17 35.32-27 22x31
36.36x27  9-13 37.45-40 13-18 *
```

An example of an analysis in PDN 3.0 format:

```
[Event "nk"]
[Site "?"]
[Date "2009.04.08"]
[Round "?"]
[White "Derkx,B."]
[Black "Meijer,Hein"]
[Result "2-0"]
[GameType "20"]
[WhiteTime "2.23"]
[BlackTime "2.40"]
[PlyCount "117"]

{ Beide spelers hebben volgens Turbo Dambase 3x eerder tegen elkaar
gespeeld. In het Nederlands kampioenschap van 2007 en 2008, en in de
halve finale van 2007, troffen beiden elkaar. Alle eerdere duels
```

(continues on next page)

(continued from previous page)

```
eindigde in remise. } 1. 34-29 17-22 2. 32-28 { Het populairste
antwoord in deze opening is 2.39-34 op ruime afstand gevolgd door
2.40-34 } 11-17 3. 37-32 6-11 4. 41-37 19-23 ( { Meestal wordt eerst
} 4... 1-6 { gespeeld om na } 5. 46-41 { alsnog } 19-23 6. 28x19
14x34 { te spelen. } ) 5. 28x19 14x34 6. 39x30 ( { Over het algemeen
is slaan met } 6. 40x29 { populairder. } ) 6... 13-19 7. 44-39 8-13
8. 50-44 20-25 9. 32-28 25x34 10. 40x29 16-21 11. 31-26 21-27
12. 37-32 11-16 13. 32x21 16x27 14. 38-32 27x38 15. 43x32 10-14
16. 49-43 5-10 17. 42-38 3-8 18. 47-42 19-23 19. 28x19 14x34
20. 39x30 13-19 21. 43-39 10-14 22. 46-41 18-23 23. 41-37 12-18
24. 37-31 7-12 25. 33-28 22x33 26. 39x28 15-20 27. 31-27 2-7
28. 44-39 7-11 29. 39-33 20-24 30. 42-37 1-6 31. 48-42 9-13 32. 30-25
23-29 33. 27-21 11-16 34. 37-31 16x27 35. 31x11 6x17 36. 45-40 17-21
37. 26x17 12x21 38. 42-37 18-23 39. 36-31 21-26 40. 31-27 8-12
41. 40-34 29x40 42. 35x44 4-10 43. 44-39 23-29 44. 28-22 13-18 ( {
Zwart had hier } 44... 12-18 { moeten spelen } 45. 22-17 29-34
46. 39x30 24x35 { enz. } ) 45. 22x13 19x8 46. 33-28 $1 8-13 47. 28-22
13-19 $2 ( { In tijdnood gaat het nodige mis, aangewezen is hier }
47... 10-15 48. 22-17 12x21 49. 27x16 13-19 { en ook zwart werkt aan
zijn doorbraak. } ) 48. 25-20 $3 ( 48. 22-17 12x21 49. 27x16 26-31
50. 37x26 24-30 51. 25x23 19x37 ) 48... 24x15 49. 22-17 12x21
50. 27x16 15-20 51. 16-11 20-24 52. 11-7 29-34 53. 39x30 24x35
54. 7-1 14-20 55. 32-27 20-24 56. 27-22 10-15 57. 1-45 24-30
58. 22-17 19-24 59. 45-18 *
```

An example of a checkers game in PDN 3.0 format:

```
[Event "Double Corner Dyke"]
[Black "Jordan,A"]
[White "Tesheliet,F"]
[Event "This is an 8x8 draughts game"]
[Result "1/2-1/2"]
[GameType "21"]

1.  9-14 22-17 2. 11-15 25-22 3. 15-19 {Forms the Double Corner Dyke, With
 black aiming to occupy sqr 19, attacking white's double corner.} 23x16
4. 12x19 24x15 5. 10x19 17x10 6.  6x15 21-17 7.  5-9 29-25 8.  8-12 25-21
9.  7-10 17-13 10.  1-6 {It seems unwise to abondon the key back row sqr 1,
 but it is necessary to prevent 13-9..} 27-24 11.  4-8 32-27 12.  9-14
27-23 13.  3-7 23x16 14. 12x19 22-17 15.  7-11 26-23 16. 19x26 30x23
17.  8-12 24-20 18. 15-18 23-19 19. 11-15 20-16 20. 15x24 28x19 21.  2-7
31-26 22. 18-23 26-22 23. 23-27 16-11 {! a really beautiful escape}
24.  7x23 22-18 *
```

An example of a live game captured with an electronic board, with clock times and two setups:

```
[White "Player 1"]
[Black "Player 2"]
[Round "1"]
1.  32-28 {[%clock w0:00:00 B0:00:05]}
    19-23 {[%clock W0:00:10 b0:00:15]}
2.  28x19 {[%clock w0:00:20 B0:00:25]}
```

(continues on next page)

(continued from previous page)

```
    14x23 {[%clock W0:00:30 b0:00:35]}
3.  34-29 {[%clock w0:00:40 B0:00:45]}
    23x34 {[%clock W0:00:50 b0:00:55]}
4.  40x29 {[%clock w0:01:00 B0:01:05]}
    10-14 {[%clock W0:01:10 b0:01:15]}
5.  37-32 {[%clock w0:01:20 B0:01:25]}
    13-19 {[%clock W0:01:30 b0:01:35]}
6.  41-37 {[%clock w0:01:40 B0:01:45]}
     8-13 {[%clock W0:01:50 b0:01:55]}
7.  46-41 {[%clock w0:02:00 B0:02:05]}
     2-8  {[%clock W0:02:10 b0:02:15]}
8.  45-40 {[%clock w0:02:20 B0:02:25]}
    17-21 {[%clock W0:02:30 b0:02:35]}
9.  31-26 {[%clock w0:02:40 B0:02:45]}
    19-23 {[%clock W0:02:50 b0:02:55]}
10. 26x17 {[%clock w0:03:00 B0:03:05]}
    23x45 {[%clock W0:03:10 b0:03:15]}
11. 36-31 {[%clock w0:03:20 B0:03:25]}
    12x21 {[%clock W0:03:30 b0:03:35]}
12. 31-26 {[%clock w0:03:40 B0:03:45]}
     7-12 {[%clock W0:03:50 b0:03:55]}
13. 26x17 {[%clock w0:04:00 B0:04:05]}
    12x21 {[%clock W0:04:10 b0:04:15]}
14. 33-29 {[%clock w0:04:20 B0:04:25]}
    21-26 {[%clock W0:04:30 b0:04:35]}
15. 41-36 {[%clock w0:04:40 B0:04:45]}
    14-19 {[%clock W0:04:50 b0:04:55]}
16. 39-33 {[%clock w0:05:00 B0:05:05]}
    20-24 {[%clock W0:05:10 b0:05:15]}
17. 29x20 {[%clock w0:05:20 B0:05:25]}
    15x24 {[%clock W0:05:30 b0:05:35]}
/FEN "W:W32,33,35,36,37,38,42,43,44,47,48,49,50:B1,3,4,5,6,8,9,11,13,16,18,26,30,45"/ {[
↪%clock W0:05:40 b0:05:45]}
19. 35x24 {[%clock w0:05:50 B0:05:55]}
/FEN "W:W32,33,36,37,38,42,43,44,47,48,49,50:B1,3,4,5,6,8,9,11,13,16,18,26,30,45"/ {[
↪%clock W0:06:00 b0:06:05]}
21. 32-28 {[%clock w0:06:10 B0:06:15]}
    30-35 {[%clock W0:06:20 b0:06:25]}
22. 43-39 {[%clock w0:06:30 B0:06:35]}
     1-7  {[%clock W0:06:40 b0:06:45]}
23. 37-32 {[%clock w0:06:50 B0:06:55]}
     7-12 {[%clock W0:07:00 b0:07:05]}
24. 49-43 {[%clock w0:07:10 B0:07:15]}
    18-22 {[%clock W0:07:20 b0:07:25]}
25. 28x17 {[%clock w0:07:30 B0:07:35]}
    12x21 {[%clock W0:07:40 b0:07:45]}
```

CHAPTER

NINE

PDN PARSING ISSUES

This section gives an overview of some issues with existing PDN definitions. In particular those issues that are important when writing a PDN parser.

9.1 Game Separator (1)

The * symbol is both used as a game terminator/separator and as a move strength indicator to denote a forced move. This introduces a nasty ambiguity. For example the string `1-6* 32-28` can be interpreted as one game containing two moves, or as two games separated by a *. Since * is commonly used in draughts publications to denote forced moves, the preferred solution would be to disallow * as a game separator, and to use a different symbol like #. However, this would completely destroy backward compatibility. A less intrusive solution is to disallow * as a move strength indicator. Note that there is an alternative available in the form of the `$7` numeric annotation glyph. Yet another solution is to demand that there can be no space between a move and it's corresponding move strength. Then a move and it's corresponding move strength can be defined as one token.

9.2 Game Separator (2)

It is common practice to terminate games with their result. In PGN this is no problem, since the chess results differ substantially from chess moves. But in draughts some results like `1-0` and `1-1` are very similar to normal draughts moves. This complicates parsing. For example, if the result `1-1` is defined as a token, then parsers may easily get confused by a move like `1-18`. Several parsers insist on parsing this as `1-1` followed by an `8`. This problem is likely to occur when a move is split up into separate tokens.

Since the result of a game can already be specified in PDN using the `Result` tag, there is no need to use a game result as a game separator. It can even be considered as bad style to have two different ways to specify the result of a game. It seems therefore logical to forbid using the result of a game as a game terminator (or separator).

9.3 Capture Separator

The squares of a capture are separated using the symbol `x`, for example in the move `32x23`. If one defines a capture as a production

> CaptureMove = Square "x" Square

then there can easily be conflicts with identifier tokens. Tokenizers are often greedy, which means that they can insist on parsing `x23` as an identifier token, instead of a capture separator `x` followed by a square `23`. Some parsers offer solutions to this type of problem, but not all of them. Note that this problem can be avoided by defining a move as a single token.

9.4 Move token

It is an important question whether a move should be defined as a single token (by means of a regular expression), or as a production consisting of multiple elements. A production has the benefit that the structure of a move can be represented more clearly. But as explained above, then a more powerful parser is needed. If a move is defined as a token, then a simple LL(1) parser is enough to parse PDN.

9.5 Move strength

In draughts publications a move strength can be wrapped in parentheses, like in `31-27(?)`. Parentheses are also used to define variations in an analysis, for example `1.32-28 18-23 2.38-32 ( 2.37-32? 23-29! ) 12-18`. This introduces an ambiguity, but in most parsers this can be resolved by defining a move strength as a single token.

CHAPTER

TEN

FEN TAG

The Forsyth-Edwards Notation or shortly FEN tag defines a position on the board. In the PDN 2.0 draft standard the following syntax is proposed:

`[TURN]:[COLOUR1][[K][SQUARE_NUM][,]...]:[COLOUR2][[K][SQUARE_NUM][,]...]`

We extend this syntax with ranges of numeric fields, such that the start position of international draughts can be defined as `[FEN "W:W31-50:B1-20"]`.

Another extension is that we allow `?` to denote an unknown color.

The following grammar describes the formal syntax of the value of a FEN tag.

```
// Productions
Fen                          : COLOR (NumericSquares | AlphaNumericSquares) DOT?
NumericSquares               : (COLON COLOR NumericSquareSequence)+
NumericSquareSequence        : NumericSquareRange (COMMA NumericSquareRange)*
NumericSquareRange           : KING? NUMSQUARE (HYPHEN NUMSQUARE)?
AlphaNumericSquares          : (COLON COLOR AlphaNumericSquareSequence)+
AlphaNumericSquareSequence : KING? ALPHASQUARE (COMMA KING? ALPHASQUARE)*

// Tokens
COLOR                        : "[WB?]"
KING                         : "K"
ALPHASQUARE                  : "[a-h][1-8]"
NUMSQUARE                    : "([1-9][0-9]*)|(0[1-9][0-9]*)|0"
HYPHEN                       : "\-"
COMMA                        : "\,"
COLON                        : "\:"
DOT                          : "\."
```

Important

Some corrections have been made.

- The `COMMA` in `NumericSquareSequence` and `AlphaNumericSquareSequence` is mandatory.
- A missing `KING?` in `AlphaNumericSquareSequence` has been added.
- `NUMSQUARE` may contain more than two digits (it is up to the implementer to do range checks).

10.1 Restrictions

The following restrictions apply when writing a FEN tag:

- No embedded spaces are allowed inside the value of a FEN tag
- No dot ('.') is allowed at the end of the value of a FEN tag

10.2 Examples

```
[FEN "B:W18,24,27,28,K10,K15:B12,16,20,K22,K25,K29"]
[FEN "B:W18,19,21,23,24,26,29,30,31,32:B1,2,3,4,6,7,9,10,11,12"]
[FEN "W:W31-50:B1-20"]
```

10.3 Extensions

The above grammar does not allow positions with no pieces for one of the players. It should therefore be extended to accept empty ranges of pieces.

CHAPTER

ELEVEN

GAMETYPE TAG

The GameType tag defines the type of the draughts game, including the size of the board and the preferred notation. It has been previously defined on [Grimminck].

The GameType tag has the following syntax:

`GameType "Type-number [,Start colour (W/B),Board width, Board height, Notation [, Invert-flag]]"`

A regular expression for the GameType tag is

```
"[0-9]+(,[WB],[0-9]+,[0-9]+,[ANS][0123](,[01])?)?"
```

The game type is a number followed by some optional attributes. Several numbers are predefined, as given by the following table. A test page for the GameType tag can be found at `gametype.html`.

Type	Game	Full details	Result type	Capture Separator
0	Chess			
1	Chinese chess			
2-19	Future chess expansion			
20	10x10 International draughts	[GameType “20,W,10,10,N2,0”]	International	x
21	English draughts	[GameType “21,B,8,8,N1,0”]	Default	x
22	Italian draughts	[GameType “22,W,8,8,N2,1”]	Default	x
23	American pool checkers	[GameType “23,B,8,8,N1,0”]	Default	x
	Pool checkers (unified) *)	[GameType “23,W,8,8,A0,0”]	Default	x
	Zimbabwean pool checkers *)	[GameType “23,W,8,8,A0,0”]	Default	x
	Jamaican draughts *)	[GameType “23,W,8,8,A1,1”]	Default	x
24	Spanish draughts	[GameType “24,W,8,8,N1,1”]	Default	x
25	Russian draughts	[GameType “25,W,8,8,A0,0”]	Default	:
26	Brazilian draughts	[GameType “26,W,8,8,A0,0”]	Default	x
27	Canadian draughts	[GameType “27,W,12,12,N2,0”]	International	x
28	Portuguese draughts	[GameType “28,W,8,8,N1,1”]	Default	x
29	Czech draughts	[GameType “29,W,8,8,A0,0”]	Default	x
30	Turkish draughts	[GameType “30,W,8,8,A0,0”]	Default	x
31	Thai draughts	[GameType “31,B,8,8,N2,0”]	Default	-
40	Frisian draughts	[GameType “40,W,10,10,N2,0”]	Default	x
41	Spantsiretti draughts	[GameType “41,W,10,8,A0,0”]	Default	:
42-49	Future draughts expansion			
50	Othello			
51..	Future expansion			

*) Note that Pool checkers and Jamaican draughts have been added later. They have the same rules as American

pool checkers, but they use algebraic notation, and white moves first. Moreover, in Jamaican draughts the direction of the numbering is vertical instead of horizontal. Zimbabwean is played on the light squares. All checkers variants share the number 23, but the abbreviated version `[GameType "23"]` still expands to `[GameType "23,B,8,8,N1,0"]` (American pool checkers).

The game types 29, 30, 31, 40 and 41 are not listed in [Wikipedia], but they were added based on conventions of the game site Play OK and the program Aurora Borealis.

Attribute	Description
Start-colour	Either W or B - white/black side starts
Board-width	Width of board.
Board-height	Height of board.
Notation	A character indicating the notation type • A = alpha/numeric like chess, • N = numeric like draughts. • S = SAN - short-form notation. followed by a number indicating the location of the first square (A1 or 1) from the perspective of the starting player • 0 = Bottom left • 1 = Bottom right • 2 = Top left • 3 = Top right
Invert-flag	• 0 = The bottom left corner is a playing square • 1 = The bottom left corner is not a playing square

Important

The interpretation of the Invert-flag has been changed! The previous interpretation (0 = Play on dark squares, 1 = Play on light squares) was not accurate for certain game types.

Note

The Start-colour field is just an indication for the colour of the pieces, but it has no influence on the notation.

Note

The principal direction of the notation is assumed to be horizontal. This means that square 2 is always to the left or to the right of square 1.

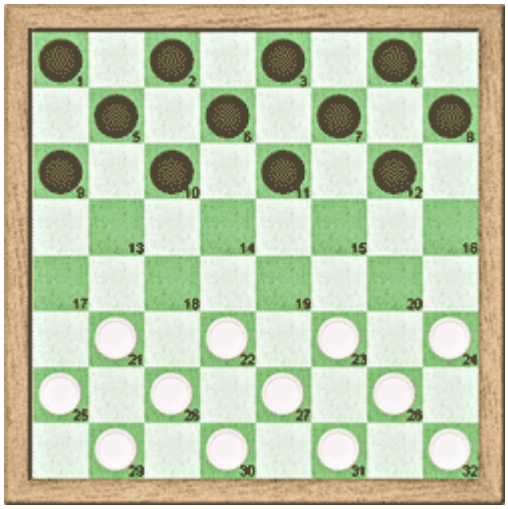
Note

Usually the game is played on the dark squares.

11.1 Examples

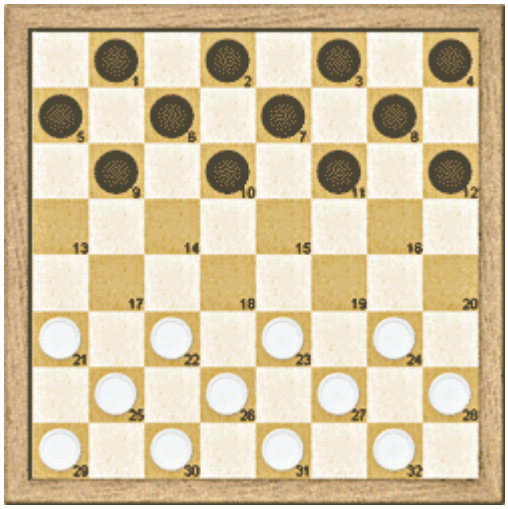
```
[GameType "0"]                  {Straight chess}
[GameType "0,W,8,8,S0"]         {Straight chess with full spec}
[GameType "20"]                 {10x10 draughts}
[GameType "21,B,8,8,N1,0"]      {English draughts with full spec}
```

Italian draughts uses the following notation:



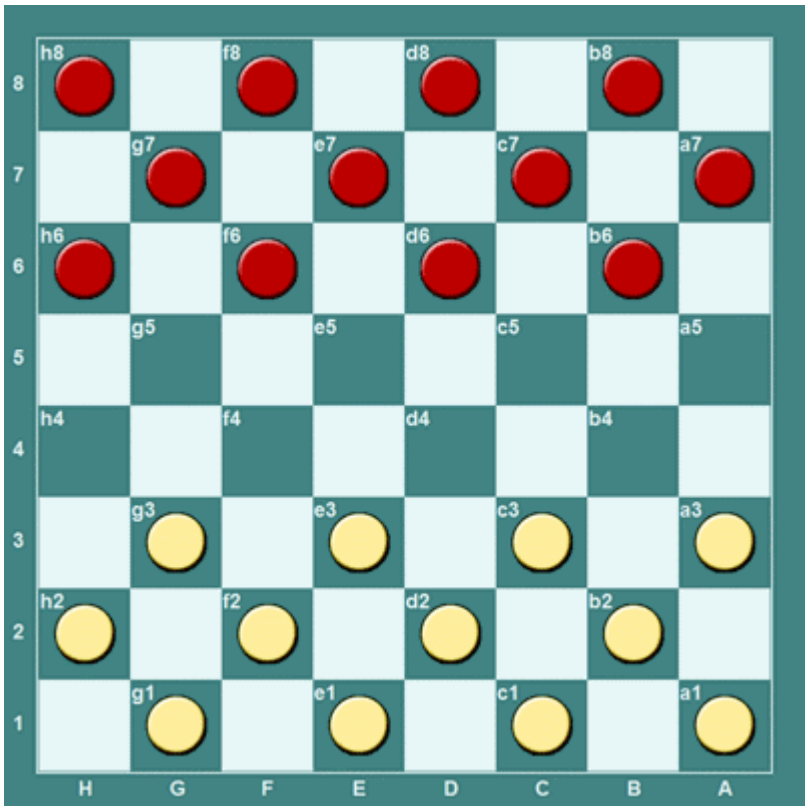
For Italian draughts the first square (1) is located in the top left corner of the board. The next square (2) can be found by moving in horizontal direction to the right. Therefore Italian draughts gets the notation number 2, that corresponds with (Top left, horizontal).

American pool checkers uses the following notation:



Pool checkers uses the following notation:

Jamaican draughts uses the following notation:



CHAPTER

TWELVE

CHANGELOG

24 June 2017

- Put the standard in a github repository on https://github.com/wiegerw/pdn.

9 June 2017

- Added some time control tag examples provided by Igor Le Masson.

1 January 2016

- Made some corrections and improvements to the FEN grammar, noted by Rein Halbersma.
- Added an archived copy of the PDN 2.0 standard, since it is no longer available online.
- Added Zimbabwean pool checkers.

3 May 2014

- Changed the GameType tag for Jamaican draughts: A5 -> A1
- Removed the vertical direction from the Notation field, since in all known cases the notation direction is horizontal.

30 April 2014

- Added a test page for the GameType tag.
- Corrected the description of the notation field of the GameType tag.
- Made some corrections to the GameType table.
  - Pool checkers (unified): A1 -> A0
  - Turkish draughts: A1 -> A0

27 April 2014

- Changed the result type of Frisian draughts to Default.

18 April 2014

- Made some corrections to the gametype table, noted by Rein Halbersma.
  - Russian/Brazilian/Czech: A1 -> A0
  - Spantsiretti: N2 -> A0
  - Thai: black starts, and N1 -> N2
- Added an improved description of move disambiguation, supplied by Ed Gilbert.
- Added a section with proposals for the next version of the standard.

27 May 2013

- Made some improvements to the gametype section, with the help of Jake Cacher.
  - Added game types Jamaican draughts and Pool checkers
  - Added some examples of the board layout of game types
  - Adapted the interpretation of the invert flag in the game type
  - Extended the notation flag of the game type, such that the Jamaican draughts notation is supported
- Added an example for setup commands, supplied by Gérard Taille.
- Added a proposal for the notation null moves, by Gérard Taille.

12 March 2012

- Added a section about character encodings.
- Adapted the allowed values for the `Result` tag, and added a `ResultFormat` tag that can be used to specify uncommon result values.
- Added `clk`, `mct`, `egt` and `emt` embedded commands.
- Added `TimeControlWhite` and `TimeControlBlack` tags for specifying individual time settings.
- Added requirements for notation type and capture separators.
- Adapted the grammars to allow alpha numeric moves without separator (a3b4).
- Added documentation about when the full capture notation may/must be used.
- Added requirement about disambiguation of ambiguous moves.
- Added a comment that the empty string is always allowed for a PDN tag value.
- Added line comments starting with a `%`.
- Added FEN values with unknown color (specified using `?`).
- Updated the grammars.
- Updated the PDN checker.

Author: Wieger Wesselink, wieger <at> 10x10 <dot> org

BIBLIOGRAPHY

[Wikipedia] Wikipedia: Portable Draughts Notation https://en.wikipedia.org/wiki/Portable_Draughts_Notation

[Nemesis] PDN 2.0 Specification by Murray Cash (draft) `pdn2.txt` An archived copy of http://www.nemesis.info/pdn2.txt, 17 July 2012.

[PGN] Portable Game Notation Specification and Implementation Guide http://www.saremba.de/chessgml/standards/pgn/pgn-complete.htm

[Sage] Adrian Millet's PDN description: https://web.archive.org/web/20111021120519/http://homepages.tcp.co.uk/~pcsol/sagehlp1.htm#PDN

[Grimminck] Michel Grimminck's PDN page https://www.xs4all.nl/~mdgsoft/draughts/pdn.html

[DGT] DGT Clock times extension https://web.archive.org/web/20100502070409/http://digitalgametechnology.com/site/index.php/Board-Driver/pgn-clock-times-extension.html

[Forum] PDN standard topic on World Draughts Forum https://damforum.nl/bb3/viewtopic.php?t=2544

[DParser] DParser, a GLR parser generator written in C http://dparser.sourceforge.net/

[Grammatica] Grammatica, an LL parser generator written in java http://grammatica.percederberg.net/

[TPG] Toy Parser Generator, a parser written in python http://cdsoft.fr/tpg/