Dot Language Specification

David Vieten

May 13, 2024

Introduction

Fortunately, I have had the privilege of being around the sport of hockey for the majority of my lifetime. Over the years, it has become clear to me that the current method of drawing face-off plays is outdated and inefficient. A face-off takes place at the start of a period or following a whistle. The ref drops the puck on a face-off dot and two center-men battle for possesion. Face-offs play an important role in hockey, as they are an opportunity to gain possession and run a set play. As a result coaches draw up routes for each of the 5 guys on the ice, dictating the route of each player on a face-off win. Often, teams have multiple versions of these set routes called ”face-off plays.” These face-off plays may change on any given week depending on who the opponent is and what their weaknesses are. As a result, the players are responsible for memorizing their routes. Currently, coaches either draw up the faceoff plays on a white board and hope the players can remember it, or they draw them up on paper and distribute photocopies for the players to study.

The programming language ”Dot” solves this problem. ”Dot” allows coaches to create face-off plays in an SVG that can be emailed to players. This language saves coaches a headache by providing a way to easily distribute face-off plays without wasting paper and ink to do so. Coaches will also be able to save face-off plays, allowing them to keep track of what plays they have used against who in the past. This ability to save face-off plays will allow coaches to save scouting reports, increasing prepardness for the team. In a typical game week, coaches will watch video on the opposing team to get an idea of what they will be faced with come game time. While watching video, coaches can create and save the opposing team’s faceoff plays to determine the best counter plays. ”Dot” will save coaches time and allow players to study their routes in an efficient manner.

Design Principles

Several design principles guide Dot’s development to address the problem of creating and distributing faceoff plays. With the goal of accessibility and usability, Dot produces an SVG allowing coaches to easily distribute visual representations of plays. Efficiency and scalability are two factors that Dot seeks to achieve. The ability to save and organize face-off plays supports scalability by allowing coaches to create libraries of strategies/pre-scouts over time. Adaptability and extensibility are also fundamental design principles in the development of the language. Dot allows for the creation of custom plays designed for specific opponents, providing an advantage in the hockey world. Furthermore, Dot’s support for saving and analyzing opponent’s face-off plays contributes to its strategic use as a tool for planning and adapting. Moreover, Dot is built with accessibility, efficiency, and adaptability in mind.

Examples

lefthash net right offense
righthash corner right offense
dot slot right offense
leftpoint walkline right offense
rightpoint halfwall right offense

stackinside corner right defense
righthash backdoor right defense
dot slot right defense
rightpoint upwall right defense
leftpoint hold right defense

lefthash hold left defense
righthash slot left defense
dot corner left defense
stackinside net left defense
stackoutside hold defense

Language Concepts

To write programs in this language, users need to have a grasp on the concepts of primitives and combining forms as well as knowledge of the sport of hockey. The primitives help define areas on the hockey rink through the use of side and zone indicators to specify the context of the location. End routes represent endpoints that players will move to from their original position once the puck drops. Combining forms involve creating these routes by pairing start and end points. These routes can be organized into a board, which is a list of routes. A sequence of strategic moves can then be represented off a faceoff. By understanding these core concepts, users can effectively design player movements that will produce an advantage on game day.

Formal Syntax

```
        <expr> ::= <route>+
        <route>::= <routedef><dotplace>
    <routedef> ::= <startroute><endroute>
    <dotplace> ::= <side><zone>
    <endroute> ::= net
                 | walkline
                 | downwall
                 | upwall
                 | corner
                 | hold
                 | slot
                 | backdoor
        <zone> ::= offense
                 |  defense
        <side> ::= right
                 |  left
  <startroute> ::= lefthash
                 |  righthash
                 |  dot
                 |  rightpoint
                 |  leftpoint
                 |  stackinside
                 |  stackoutside
```

Semantics

Table 1: Semantics of Language

Syntax	Abstract Syntax	Prec./Assoc.	Meaning
<side>	Side of string	N/A	Side is primitive. Represents the sides of the ice: ”right” or ”left”.
<zone>	Zone of string	N/A	Zone is primitive. Represents the zones of teh ice where players will line up.: ”offense” or ”defense”.
<route>	Route of RouteDef * DotPlace	N/A	Route is a combing form. Represents a path or route of a player on the ice, defined by a route definition and a dot placement.
<routedef>	RouteDef of StartRoute * EndRoute	N/A	routedef is a combining form. Represents the definition of a route, consisting of a start point and an end point. Dictates the coordinates of startroute and endroute.
<dotplace>	DotPlace of Side * Zone	N/A	dotplace is combining form. Represents the placement of a dot along a route, defined by side and zone.
<startroute>	StartRoute of string	N/A	startroute is a primitive. Represents the possible starting points or areas of a route. Eval turns these into coordinates.
<endroute>	EndRoute of string	N/A	endroute is a primitive. Represents the possible ending points of a route. Eval turns these into coordinates.

Remaining Work

Right now, Dot is capable of producing routes for a singular team either on the offensive or defensive side of the dot. The next step to further inprove this langauge is to incorperate the ability to show players skating with or passing the puck. This requires drawing dotted and squiggly lines as opposed to stright lines. By doing this, players would know their routes of the face-off, as well as the where the puck will be. Given our set time period, I am unable to implement this feature. With some more experience in SVG, I think this feature would greatly enhance the abilities of Dot.