An Offline Neuro-Symbolic Football Pattern Retrieval Approach Using Constraint Programming

Augustin Crespin
ICTEAM, UCLouvain, Belgium

Pierre Schaus
ICTEAM, UCLouvain, Belgium

Abstract

Using a single broadcast camera, modern deep learning methods can detect and label players and ball positions on a frame-by-frame basis. This work focuses on post-game analysis, where frame-level labels are available for the entire video sequence. Deep learning alone performs poorly when retrieving intervals of frames in which specific spatio-temporal conditions or tactical patterns occur involving players and ball positions. A loosely coupled neuro-symbolic approach is proposed, in which these precomputed frame-level detections are processed through an SQL-like domain-specific query language. Each query is compiled into a Constraint Programming (CP) model that retrieves intervals of frames satisfying the specified constraints. The method leverages well-established CP constructs, such as time intervals and regular constraints. Experiments on real football games demonstrate that this approach is simple and efficient, enabling expressive querying for post-game tactical analysis while remaining accurate and scalable.

2012 ACM Subject Classification Software and its engineering → Constraint and logic languages; Information systems → Video search; Theory of computation → Constraint and logic programming; Theory of computation → Pattern matching

Keywords and phrases Pattern Matching, Domain-Specific Language, Video Analysis

Digital Object Identifier 10.4230/LIPIcs.CP.2026.45

Supplementary Material *Software (Source Code)*: `https://github.com/crespina/TactiCP`

1 Introduction

When preparing for upcoming matches, football team coaches must create a well-defined game-plan. The production of this plan will most likely require watching hours of the opponents' previous matches in order to analyze what tactics they employ, a costly bottleneck. In professional leagues, coaches even have multiple people at their disposal doing video analysis as a full-time job. Their work is tedious because the video analyst does not necessarily know beforehand what to look for. They often rely on high-level statistics such as the number of passes, ball possession percentage, etc. However, they could benefit from being able to look for specific tactics. Post-game analysis is also a very important part of a coach's job. For example, when trying to implement new strategies, it is crucial to verify how many and how those strategies have been used during the match. However, being able to search for these tactics in the video feed instead of re-watching the whole clip looking out for a specific sequence of events would significantly improve efficiency. We refer to [17] for a recent and complete review of the topic.

With the rise of multimodal large language models, one might be tempted to use one of these video understanding models for question answering. We tried giving a 30-second clip of a football game to VideoLLaMA 3-7B [30], a state-of-the-art multimodal foundation model designed to excel in both image and video understanding tasks. The clip contains six passes, but, when prompted with "How many passes can you count?", the model gave

incorrect and inconsistent counts (10, 14, at least 5, and even hallucinated a commentator stating there are 40 passes). This example illustrates that purely neural models designed for video understanding are not yet reliable for more advanced tasks such as tactic detection in sports, even when the tactic in question is fairly simple (e.g., a single pass). This highlights the need for a dedicated system. These experiments motivated the development of a hybrid neuro-symbolic approach using CP based on annotations produced by machine learning (ML) methods.

1.1 Our Approach: TactiCP

In this paper, we propose a loosely-coupled neuro-symbolic approach, illustrated in Figure 1. The dataflow relies on three distinct processing units:

1. **ML Tracking (Neural):** The raw broadcast video is fed frame-by-frame into an ML tracking model, which extracts the continuous 2D positions of the players and the ball.
2. **Possession Extraction (Heuristic):** The raw positional data is passed to a heuristic module that determines which player, if any, possesses the ball at each individual frame.
3. **CP Compilation & Solving (Symbolic):** The user expresses a tactical pattern query via a Domain-Specific Language (SQL-like DSL syntax). Based on this query and the discrete data (positions and possession), a CP model is compiled and solved to retrieve the frame interval(s) satisfying the query constraints.

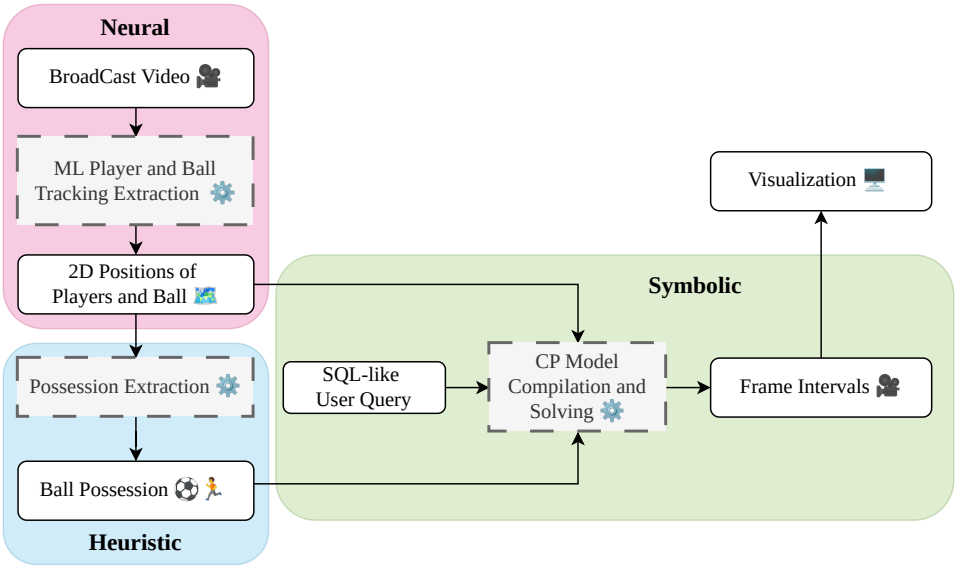


Figure 1 Dataflow: Dashed rectangular nodes represent processing units (neural, heuristic, and symbolic), while rounded nodes denote input/output data artifacts.

This paper focuses exclusively on the **Heuristic** and **Symbolic** components of the pipeline. The first stage, namely the Game State Reconstruction (GSR), is assumed to be completed beforehand and is therefore not addressed in this work. For the remainder of the paper, we rely on the ground-truth annotations from the SoccerNet GSR task [24].

To intuitively understand the system, consider a user who wants to find, in a game video, all sub-video intervals where Player 11 passes the ball to Player 16. Furthermore, these intervals should start exactly when the ball leaves the foot of Player 11 and end when it arrives at Player 16. The user could formulate the following simple query:

Listing 1 Pass from Player 11 to Player 16 query

```
SELECT(PLAYER("player11",11).PASSTO(PLAYER("player16",16)).MINRANGE())
```

Under the hood, our proposed approach named TactiCP translates a high-level query into a Constraint Satisfaction Problem (CSP), whose solutions correspond to one or more intervals of video frames that satisfy the query constraints. To better understand the capabilities of the approach and to experience the computation in real time through example queries and their outputs, we invite interested readers to interact with the online tool available at `https://tacticp.com`.

Our contributions are thus threefold:

1. an SQL-like Domain Specific Language for spatio-temporal tactics;
2. a compilation technique into a CP model relying on the known interval variables [16] and the regular constraint [20] adapted for interval variables;
3. an open-source tool and experiments.

We provide the repository of the source code **here**.

The paper is organized as follows. Section 2 presents the related work. Section 3 introduces the intuitive building blocks of our DSL (with the formal grammar relocated to Appendix A). Section 4 outlines the methodology to transform a query into a CP model. Section 5 conducts experiments to validate the tool's capacity to retrieve relevant patterns. Section 6 describes the tool's user interface. Finally, Section 7 concludes the paper.

2 Related work

The GSR task, corresponding to the neural step of the dataflow in Figure 1, is a recent research area. [24] introduced the first end-to-end pipeline reconstructing football game states as minimap representations. This task involves several components such as player detection and tracking, pitch localization, team recognition, and jersey number identification. The authors released a dataset together with the SoccerNet GSR Challenge. Subsequent winners of the 2024 and 2025 editions proposed improved pipelines [10, 19].

A closely related line of work is action spotting, which localizes football actions in video. Surveys summarize recent advances in sports video understanding and spotting methods [8, 28]. SoccerNet introduced a benchmark and challenge for this task [7]. Most approaches rely on ML techniques, including convolutional neural network (CNN) architectures such as NetVLAD [3, 9], transformers [31], "You Only Look Once" (YOLO) detectors [14], or dense detection anchors [23]. Algorithmic alternatives also exist; for instance, [26] detects events using a possession heuristic based on distances between players and the ball, similar to our approach.

Other studies focus on tactical pattern analysis. Passing strategies can be extracted using Dynamic Time Warping (DTW) on pass coordinates [13], or by modeling pass sequences as documents and applying text-mining techniques to identify collaborative patterns between players [27].

While most prior work either detects events from video or mines patterns offline, few systems allow analysts to interactively query spatio-temporal patterns in reconstructed game states. The closest work is [25], where users define patterns visually on a field representation and search for similar sequences in a database. However, it relies on proprietary data where event detection and 2D reconstruction are performed beforehand.

Within CP, related work mainly concerns declarative pattern mining. Mining tasks can be expressed as constraint satisfaction or optimization problems, as illustrated by MiningZinc [11] and earlier CP models for itemset mining [12]. These models were later improved through specialized global constraints such as CoverSize [21], and extended to tasks like association rule mining [4]. CP has also been applied to frequent sequence mining, with early models [18] improved through specialized constraints [15, 1, 2], including time-constrained patterns. Frequent pattern mining has also been applied to football analytics outside CP [6], where phases of play are clustered and sequential patterns mined using SPADE [29]. Another related direction is episode mining in long event sequences, addressed in CP by the EpisodeSupport constraint [5]. These works highlight the suitability of CP for modeling complex constraints over sequential data. Building on these declarative approaches, TactiCP applies CP to querying spatio-temporal patterns in reconstructed football game states.

3 DSL Overview

We introduce a DSL with a syntax inspired by SQL. It allows for the description of complex spatio-temporal pattern queries by combining atomic events with logical operators and spatial or temporal constraints.

A query in TactiCP typically follows a `SELECT ... WHERE ... FROM ...` structure. The `SELECT` clause specifies the sequence of events to retrieve. The result of a query is a solution consisting of a sequence of frame intervals $S = \langle I_1, I_2, \ldots, I_k \rangle$. Each interval I_i is defined by its start frame $start(I_i)$, its end frame $end(I_i)$, and its duration $dur(I_i) = end(I_i) - start(I_i)$. It represents the half-open temporal window $[start(I_i), end(I_i))$, where the frame at $end(I_i)$ is not included. The intervals are not necessarily contiguous, do not overlap in time and occur in chronological order: $\forall i \in \{1, \ldots, k-1\} : \quad end(I_i) \leq start(I_{i+1})$. Similarly to a CP solver, one can search either for all solutions or m solutions. When multiple solutions are produced, they are returned in lexicographic order with respect to time.
To illustrate the capacities of the DSL and its core concepts, consider the following examples of increasing complexity:

▶ **Example 1** (Basic Movement).

```
SELECT(BALL().MOVETO(6, 7).MINRANGE())
```

A solution to this query is a single frame interval $S = \langle I_1 \rangle$ where the ball moves from zone 6 to zone 7 (a representation of the zones is shown on Figure 2). The `MINRANGE()` modifier is a fundamental concept: it instructs the solver to find the *minimal* contiguous interval where the predicate holds, rather than the largest one.

▶ **Example 2** (Logical Combinations).

```
SELECT(
    AND(
        PLAYER("p11", 11).PASSTO(PLAYER("wc1")).MINRANGE(),
        NOT(POSITION(BALL(), 1).MINRANGE())
    )
)
```

A solution to this query is a single frame interval $S = \langle I_1 \rangle$. The `AND` operator requires both conditions to be true simultaneously on the interval: player with ID 11 must be passing the ball to another player (a wildcard), and during this pass, the ball must *not* be inside of zone 1 (using the `NOT` operator).

▶ **Example 3** (Temporal & Spatial Constraints).

```
SELECT(
    POSSESSION(PLAYER("p1")).WHERE(RADIUS(-25, 10, 15)),
    PLAYER("p2",13).MOVETO(1, 6).MINRANGE().WHERE(ENDMAX(400))
)
.WHERE(MAXDURATION(100), STARTMIN(100))
```

A solution to this query is a sequence of two frame intervals $S = \langle I_1, I_2 \rangle$. The `WHERE` clause enables users to specify constraints on the sequence of intervals globally or on each event individually. For I_1, a player possessing the ball is constrained to lie within a spatial radius of 15 meters centered at $(-25, 10)$. For I_2, player 13 moving from zone 1 to zone 6 must end before frame 400. Recall that in any valid solution, I_2 occurs after I_1. The total duration, expressed as the number of frames spanned by the two retrieved intervals, is constrained by `MAXDURATION(100)`. Formally, $\mathit{end}(I_2) - \mathit{start}(I_1) \leq 100$. The constraint `STARTMIN(100)` ensures that the entire sequence starts after frame 100, i.e., $\mathit{start}(I_1) \geq 100$.

▶ **Example 4** (Complex Sequences).

```
SELECT(
    AND(
        PLAYER("p11", 11).PASSTO(PLAYER("wc1")).MINRANGE(),
        NOT(POSITION(BALL(), 1).MINRANGE())
    ),
    TEAM("left").MOVETO(3, 4, 2),
    PLAYER("p11").PASSTO(PLAYER("p12",12)),
    PLAYER("p12").PASSTO(PLAYER("p11"))
)
```

A solution to this query is a sequence of four frame intervals $S = \langle I_1, I_2, I_3, I_4 \rangle$. I_1 corresponds to the complex pass described in Example 2. I_2 is an interval during which at least two players from the left team move from zone 3 to zone 4. I_3 and I_4 represent a "one-two" between players 11 and 12: player 11 passes to player 12 during I_3, and player 12 passes back to player 11 during I_4. The sequence constraint ensures that $\mathrm{end}(I_1) < \mathrm{start}(I_2)$, $\mathrm{end}(I_2) < \mathrm{start}(I_3)$, and $\mathrm{end}(I_3) < \mathrm{start}(I_4)$.

The DSL also supports specifying the source data with `FROM` and restricting the number of solutions returned with `SEARCH`. The complete formal grammar and syntax rules defining the DSL are detailed in Appendix A.

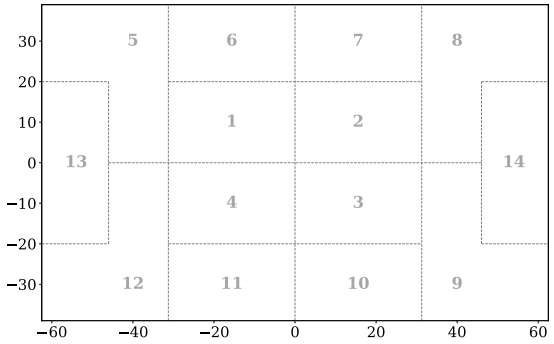


Figure 2 Zones segmentation of the field

4 Methodology

The methodology of TactiCP follows a two-stage process. First, the raw tracking data from video analysis is preprocessed to extract high-level semantic information, specifically the spatial zones of players and ball possession at each frame. Second, these extracted sequences are used as input for a CP model that implements the DSL query logic to retrieve the requested patterns. The following subsections detail these two stages.

4.1 Possession Extraction and Zone Extraction

To execute a query, TactiCP requires three key pieces of information for each frame: the positions of all players, the position of the ball, and which player (if any) is in possession of the ball. While deep-learning-based video analysis provides raw (x, y) coordinates, this data must be transformed into a format suitable for the CP solver. Specifically, we compute two primary symbolic sequences: $\mathcal{Z}$ and $\mathcal{P}$.

The first sequence, $\mathcal{Z}$, maps each player and the ball to a discrete zone on the pitch at every frame. The field is segmented into a grid of zones, as illustrated in Figure 2. Assigning a coordinate to a zone is a direct geometric mapping.

The second sequence, $\mathcal{P}$, identifies the player in possession of the ball at each frame f. Determining possession from raw trajectories is challenging because a player does not always have the ball at their feet (e.g., during a dribble or a pass). We employ a heuristic that combines the relative distance between players and the ball with the *ball activity*. The intuition is that if the ball suddenly changes direction or accelerates (high activity), it is likely being kicked or deflected by the player closest to it, even if they are not perfectly contiguous with the ball.

Let $\mathbf{x}_p^{(f)}$ and $\mathbf{x}_b^{(f)}$ be the positions of player p and the ball b at frame f, respectively. We define the ball activity score σ_f as a weighted combination of its scalar acceleration and its angular change in direction:

$$\sigma_f = w_{acc} \cdot \|\mathbf{a}_b^{(f)}\| + w_\theta \cdot |\Delta\theta_b^{(f)}| \tag{1}$$

where $\mathbf{a}_b^{(f)}$ is the ball's acceleration vector, computed from finite differences of the ball velocities derived from consecutive positions, and $\Delta\theta_b^{(f)}$ is the change in its velocity vector angle, i.e. the absolute change in direction between consecutive velocity vectors.

High values of σ_f indicate a "kick event." The possession $P(f)$ is then assigned to the closest player p^* if they satisfy distance thresholds that are relaxed when ball activity is high.

Algorithm 1 formalizes this logic. It takes as input the tracked positions, the pre-computed ball activity score σ, a distance threshold τ_d (set to 0.058 in normalized pitch coordinates) and an activity threshold τ_a (set to 47). The heuristic considers three conditions for assigning possession: (1) high ball activity near a player, (2) very close proximity, or (3) sufficient proximity and clear isolation from other players.

Algorithm 1 Ball Possession Detection

Input: Positions X, Ball activity score σ, Thresholds τ_a, τ_d
Output: Possession sequence $\mathcal{P}$

foreach *frame* f **do**
  $\mathbf{x}_b \leftarrow$ ball position;
  $p^* \leftarrow \text{argmin}_p \|\mathbf{x}_p^{(f)} - \mathbf{x}_b^{(f)}\|_2$;
  $d^* \leftarrow \min_p \|\mathbf{x}_p^{(f)} - \mathbf{x}_b^{(f)}\|_2$;
  $d^{**} \leftarrow$ second smallest distance to the ball;
  if $\sigma^{(f)} > \tau_a$ ***and*** $d^* < 3\tau_d$ **then**
    $\mathcal{P}[f] \leftarrow p^*$; *// Active ball near player*
  else if $d^* < \tau_d$ **then**
    $\mathcal{P}[f] \leftarrow p^*$; *// Close proximity*
  else if $d^* < 2\tau_d$ ***and*** $d^{**} > 4 \cdot \tau_d$ **then**
    $\mathcal{P}[f] \leftarrow p^*$; *// Close and isolated*
  else
    $\mathcal{P}[f] \leftarrow \emptyset$;
return $\mathcal{P}$

4.2 CP Model

To implement the query logic, TacticCP maps the sequence of events defined in the `SELECT` clause into a set of constraints over interval variables [16]. The core of this mapping is a specialized constraint that ensures a temporal interval matches a symbolic pattern. This is referred to as the *Substring-Regular* constraint. While the classical *regular* constraint [20] enforces that a fixed-length array of variables forms a word accepted by a deterministic finite automaton (DFA), it is not directly applicable to the current setting for two reasons. First, the sequence of interest is not fixed in advance; the relevant portion of the data is determined dynamically by the start and end of an interval variable. Second, the elements of the sequence are fixed data (e.g., pre-extracted zones $\mathcal{Z}$ or possession $\mathcal{P}$), not decision variables; the only decision variables are the bounds of the interval itself.

To capture this requirement, the *Substring-Regular* constraint is formalized as follows. Let $S = \langle s_1, \ldots, s_m \rangle$ be a fixed sequence of symbols over an alphabet Σ. Let I be an interval variable with start $start(I)$, end $end(I)$, and duration $dur(I) = end(I) - start(I)$. The interval defines a substring $S[start(I), end(I) - 1]$.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA where Q is the finite set of states, Σ is the input alphabet, δ is the transition function, q_0 the initial state and F the accepting states, and $\mathcal{L}(A)$ is the language the DFA recognizes.

▶ **Definition 5.** *The constraint SubstringRegular(S, I, A) holds if and only if the substring selected by I is accepted by A, i.e., $S[start(I) \dots end(I) - 1] \in \mathcal{L}(A)$. Equivalently, there exists a sequence of states $q_0, q_1, \dots, q_{dur(I)}$ such that:*

$$q_0 = q_0, \quad q_{k+1} = \delta\big(q_k, s_{start(I)+k}\big) \text{ for } k \in [0, dur(I) - 1], \quad q_{dur(I)} \in F.$$

The filtering algorithm for this constraint is detailed in Section 4.3.

To illustrate how the system uses this constraint, the compilation of the CP model is analyzed for each of the example queries given in Section 3.

▶ **Example 1** (Basic Movement).

Algorithm 2 Ball Movement Model

Input: Zones $\mathcal{Z}$, Ball b, Start zone z_s, End zone z_e
Output: Interval variable I matching the movement
1 $I \leftarrow$ new interval variable;
2 $regex \leftarrow (z_s)\,(\neg z_e)^*\,(z_e)$;
3 $A \leftarrow \text{DFA}(regex)$;
4 post SubstringRegular($\mathcal{Z}[b]$, I, A);
5 **return** I

Algorithm 2 describes the steps to compile the query 1 into a CP model. The process begins by creating an interval variable I for the ball movement event. Next, a regular expression describing the event is defined. In this case, the movement consists of three steps: first, the ball is in the start zone, may go through other zone(s), and finally reaches the end zone. Since the event uses the `MINRANGE` modifier, the last frame during which the ball is in the start zone and the first when it reaches the end zone are selected. Finally, a SubstringRegular constraint is posted. It ensures that the substring of $\mathcal{Z}[b]$ defined by the bounds of I is accepted by the automaton A.

▶ **Example 2** (Logical Combinations).

Algorithm 3 Logical Combinations model

Input: Possession $\mathcal{P}$, Zones $\mathcal{Z}$, Passer p_1, Ball b, Accepted zones Z
Output: Interval variable I matching the complex event
1 $I \leftarrow$ new interval variable ; *// Intersection of child events*
// Pass modeling
2 $regex_{pass} \leftarrow (p_1)\,(0)^*\,(p_{wc})$;
3 post SubstringRegular($\mathcal{P}$, I, DFA($regex_{pass}$));
// Ball position (negation)
4 $regex_{move} \leftarrow (\neg Z)^+$;
5 post SubstringRegular($\mathcal{Z}[b]$, I, DFA($regex_{move}$));
6 **return** I

The `AND` operator groups several events. An interval variable is created to represent the parent event, defined as the intersection of the child intervals. Algorithm 3 compiles the query 2. The passer is identified as player 11, while the receiver is a wildcard p_{wc} (any teammate). A pass is defined as a sequence on the `possession` sequence $\mathcal{P}$ where player A has the ball, followed by a gap with no possession, and ending with player B in possession. This corresponds to the regex $(A)^+(0)^*(B)^+$. Since the ball's position is negated in the query, the intervals during which the ball is located in any zone *not* in the set Z are identified. Both conditions are enforced using SubstringRegular constraints.

▶ **Example 3** (Temporal & Spatial Constraints).

Algorithm 4 Temporal & Spatial model

Input: Possession $\mathcal{P}$, Zones $\mathcal{Z}$, Valid possessor IDs $vIds$, Moving player ID pl_{id}, Start zone z_s, End zone z_e, Bounds $e_{max}, d_{max}, s_{min}$, Circle (c, r)
Output: List of interval variables $\mathbf{I}$

// Possession event
$I_1 \leftarrow$ new interval variable;
$regex_{poss} \leftarrow (\neg vIds)\,(vIds)^+\,(\neg vIds)$;
post SubstringRegular($\mathcal{P}$, I_1, DFA($regex_{poss}$));
$B_{in} \leftarrow$ boolean array where $B_{in}[f] = (\|\mathbf{x}_b^{(f)} - c\| \leq r)$;
post SubstringRegular(B_{in}, I_1, DFA(True$^+$)) *// RADIUS constraint*

// Move event
$I_2 \leftarrow$ new interval variable;
$regex_{move} \leftarrow (z_s)\,(\neg z_e)^*\,(z_e)$;
post SubstringRegular($\mathcal{Z}[pl_{id}], I_2$, DFA($regex_{move}$));
post end(I_2) $\leq e_{max}$ *// ENDMAX*

// Global sequence constraints
post end(I_2) − start(I_1) $\leq d_{max}$ *// MAXDURATION*
post start(I_1) $\geq s_{min}$ *// STARTMIN*
return $\langle I_1, I_2 \rangle$

Algorithm 4 compiles the query 3, which involves multiple events. For the possession event I_1, the `RADIUS` constraint is enforced by pre-computing a boolean array B_{in} indicating whether the ball is within distance r of the center c. A SubstringRegular constraint is then posted requiring I_1 to be a "true" substring of B_{in}. The movement event I_2 is modeled similarly to the basic movement example, with an additional boundary constraint `ENDMAX` on its end time. The total sequence is constrained by `MAXDURATION` and `STARTMIN`, which apply to the full span $[start(I_1), end(I_2)]$.

4.3 Filtering the Substring-Regular Constraint

A simple bound-consistent filtering for the *Substring-Regular* constraint (Definition 5) is provided. This filtering is triggered by updates to the domains of the start, end, or duration interval variables. A layered directed acyclic graph (DAG) is constructed representing the product of the sequence positions and the automaton states. The graph has $n + 1$ layers, where each layer $i \in \{1, \ldots, n + 1\}$ contains nodes (i, q) for $q \in Q$. Layer i corresponds to the position just before reading symbol s_i (with layer $n + 1$ after the last symbol).

The filtering proceeds in three phases:

1. **Forward Pass:** For every possible start position $s \in \text{dom}(\textit{start}(I))$, node (s, q_0) is marked as forward-reachable. A sweep is then performed from left to right. For each forward-reachable node (i, q) with $i \leq n$, the transition $\delta(q, s_i) = q'$ makes node $(i+1, q')$ forward-reachable.
2. **Backward Pass:** For every possible end position $e \in \text{dom}(\textit{end}(I))$ and every final state $q_f \in F$, node (e, q_f) is marked as backward-reachable. A sweep is performed from right to left. A node (i, q) with $i \leq n$ is backward-reachable if its successor $(i+1, \delta(q, s_i))$ is backward-reachable.
3. **Pruning and Bound Computation:** A node (i, q) remains feasible if it is both forward- and backward-reachable. The domain of $\textit{start}(I)$ is pruned by removing any s such that (s, q_0) is not both forward- and backward-reachable. Similarly, the domain of $\textit{end}(I)$ is pruned by removing any e such that no node (e, q_f) with $q_f \in F$ is both forward- and backward-reachable.

The complexity is $\mathcal{O}(n \times |Q|)$, where n is the number of frames and $|Q|$ is the number of states in the automaton. In our application, n is typically 750 (for a 30-second clip) but can reach 67,500 for a whole half-time, and $|Q|$ is small (e.g., 24 states for a pass model), leading to very efficient filtering in practice.

5 Experiments

The experimental section aims to answer the following questions:

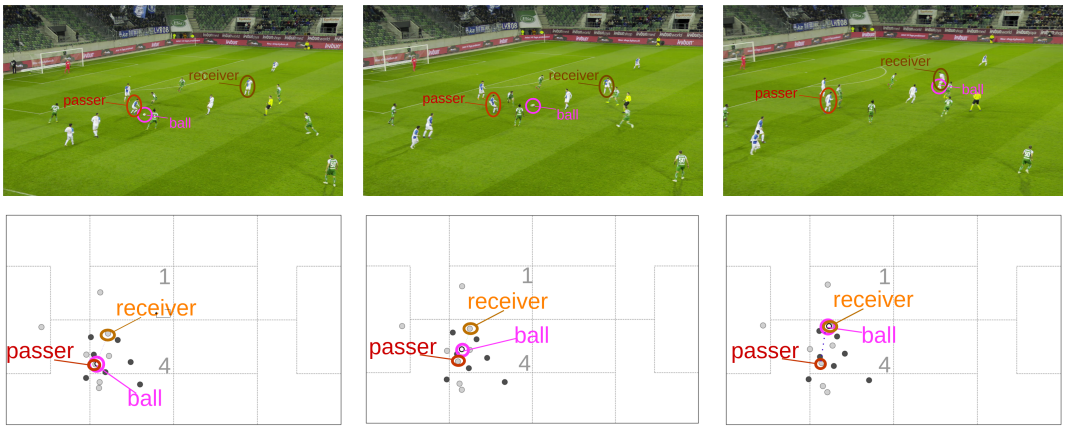
1. Is the system accurate?
2. Is the system scalable to full length videos?
3. Can we use the system to do analytics, a data-mining task?

Experiments on TactiCP were conducted on a virtual machine with 1 vCPU and 32 GB of RAM, running Ubuntu 24.04. The *Substring-Regular* constraint was implemented in MaxiCP [22] which supports interval variables. The dataset used consists of the ground-truth annotations provided for training models for the GSR task of the 2024 SoccerNet challenge[1]. The data are stored in JSON format and include, among other information, the 2D reconstructed positions of each player and the ball for every frame of the corresponding videos. Each JSON file corresponds to one of the 164 available 30-second clips.

5.1 Accuracy

The accuracy of TactiCP is evaluated through both qualitative and quantitative lenses. First, we perform a qualitative assessment by executing complex queries and manually verifying the correctness of the retrieved intervals. Due to the lack of exhaustive ground truth annotations for all possible tactical patterns, we provide illustrative examples that allow the reader to judge the relevance of the results. Second, we conduct a quantitative comparison for the specific task of pass detection, using the predictions of a state-of-the-art ML model as a reference.

[1] The rules and development kit for the GSR challenge can be found **here**



(a) Start (frame 84) **(b)** Middle (frame 100) **(c)** End (frame 117)

Figure 3 Illustration of the AND event. The top row shows frames from the original broadcast video, while the bottom row presents the corresponding spatial abstraction used by the system. From left to right, the frames correspond to the start of the pass, an intermediate moment during the pass, and the end of the pass. During all this interval, the ball never goes to zone 1.

5.1.1 Qualitative evaluation

In this task, defining a quantitative accuracy metric for any arbitrary query is inherently difficult. For example, whether an event qualifies as a pass is often subjective, and even human annotators may disagree due to ambiguous situations (e.g., distinguishing between a pass, a lost ball, or a shot). This makes establishing a definitive ground truth for every pattern problematic. Instead, we showcase representative queries reflecting realistic use cases. We only illustrate the textual and visual output example 2 query. Others can be found in Section B or can be visualized on our online tool at `www.tacticp.com`.

▶ **Example 2** (Logical Combinations).
When given the query 2 and an instance of the dataset (here "SNGS-061"), the engine returns one solution equal to the interval during which player 11 passes and the ball is not in zone 1. The textual and visual results are displayed in Listing 2 and Figure 3 respectively.

Listing 2 Example 2 textual solution

```
THE NEXT 2 EVENTS ARE PART OF THE "AND" |
frames: start = 84 length = 33 end = 117

  EVENT # 1 PASS From player ID 11 to player ID 14 |
  frames: start = 84 length = 33 end = 117

  EVENT # 2 BALL IS NOT IN ZONE 1 |
  frames: start = 0 length = 128 end = 128
```

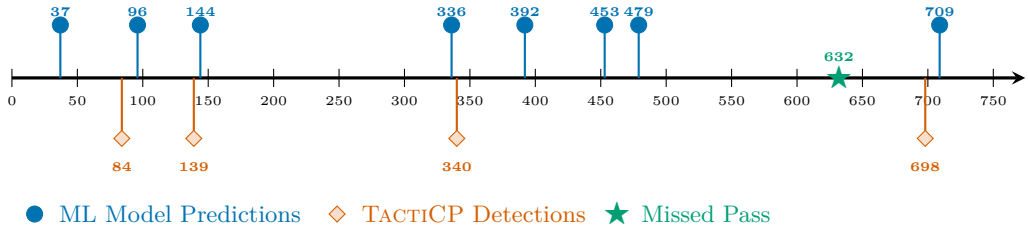


Figure 4 Start frames of passes detected by both methods on clip SNGS-061

5.1.2 Pass Detection Comparison

The SoccerNet Ball Action Spotting Challenge[2] consists of identifying ball-related actions in full football game videos across 12 classes. We compare our performance in pass detection with the winner of the 2023 edition of the challenge[3].

The ML model uses a slow-fusion architecture. In the early stages, 2D convolutions independently process input images and produce visual features. In the later stages, 3D convolutions process these visual features and produce temporal features. During inference, the model slides a 15-frame temporal window over the video and predicts whether a pass occurs at the center frame. Both methods were executed on the same machine. The ML model was run on an NVIDIA L40S GPU limited to 12GB of vRAM, whereas our approach runs on the CPU and does not require hardware acceleration.

Since no frame-level ground truth annotations are available for passes in this dataset, we cannot directly compute standard evaluation metrics against human labels. Instead, we compare the detections produced by our method with those of the winning model of the 2023 challenge and analyze their agreement.

The ML model detected 1164 passes, while our approach detected 449. Among these detections, 289 correspond between the two methods (two detections are considered a match if their start frames differ by at most 13 frames, approximately 0.5 seconds at 25 frames per second). Figure 4 shows the start frames of the passes detected by both methods on clip SNGS-061, which is used throughout the paper. The video can be viewed in the web application to assess the quality of the detections.

At frame 37, the player from the green team loses possession of the ball, making this detection from the ML model a false positive. Frames 84 and 96, 139 and 144, and 336 and 340 form three pairs (each pair being less than 13 frames apart), and all three correspond to actual passes. The ML model also predicts a pass at frame 392, but the images show a shot, making this another false positive. At frame 453, the goalkeeper makes a save from the preceding shot and the defender then retrieves the ball. Although debatable, we argue this should not be considered a pass because of the absence of intent and control over the ball trajectory. At frame 479, the player kicked into touch, another false positive for the ML model. At frame 632, a throw-in occurs. This should be considered a pass, but neither method detects it, as it comes from outside the field and is not executed with the feet. Finally,

[2] The rules and development kit for this challenge can be found **here**
[3] The source code can be found **here**

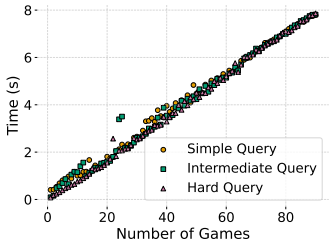


Figure 5 Execution times Games are gradually added

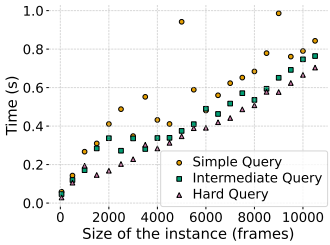


Figure 6 Execution times Bigger instance sizes

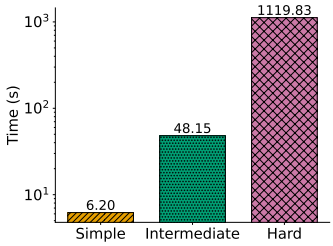


Figure 7 Execution time Half-time (67,500 frames)

frames 698 and 709 form a pair between the two methods and correspond to a genuine pass in the video.

Overall, both models correctly identify four out of five passes, however our approach produces no false positives while the ML model generates four false positives. On this example clip, our precision is thus 1.0, the recall is 0.8 and the F1 score is 0.89. The ML model achieves a precision score of 0.5, a recall of 0.8 and a F1 score of 0.62. It illustrates the typical behavior of the two methods: the ML model produces more detections but also more false positives, while our approach generates fewer but more precise detections.

Across the 164 clips, the ML model required on average 7.82 seconds per clip (std = 0.066 s), while our approach required 0.095 seconds (std = 0.0186 s), making it roughly two orders of magnitude faster.

5.2 Scalability

For the scalability experiments, we used three queries of increasing complexity, representing easy, intermediate, and hard scenarios. Those queries are introduced in Section 3.

The first experimental setting consists of incrementally adding games in the `FROM` clause of a query, as described in Section A.2. The results are displayed on Figure 5. The growth in execution time is linear with the number of games added, as expected. Indeed, when a new game is added to the `FROM` clause, the system treats it as a new instance and simply creates a new model, and starts execution from scratch.

Since we do not have access to ground-truth instances that are longer than thirty seconds, we construct larger instances by concatenating smaller ones. Consequently, the results are not intended to be meaningful, as the objective of this experiment is solely to measure execution times rather than accuracy.

The results are shown on Figure 6. The execution times follow a quadratic growth. A least-squares quadratic fit yields coefficient between 3.30×10^{-10} and 2.88×10^{-7} depending on the difficulty of the query. Figure 7 presents execution time for a full half (i.e. 45 minutes, or 67,500 frames). For simple and intermediate queries the runtime remains acceptable (<2% of the total length of the video, under the minute), whereas the most complex pattern requires markedly more time (≈18 minutes) on our experimental setup. In such cases we recommend processing long clips in smaller segments. Splitting at natural breaks (e.g. stoppage time) reduces peak runtime while preserving detectability.

5.3 Analytics

It is possible to count the number of times a pattern occurs with TactiCP. For instance, the query described in Listing 3 will count all the passes from the team on the left of the field during the whole game. In this case, the query returns an integer representing the total count of solutions found, i.e. the number of pattern matched. When launched on the first instance of the dataset ("SNGS-060"), the output will be `"5 solutions matched"`: there are five passes between members of the left team in the clip.

Listing 3 Counting the number of passes from the left team

```
SELECT(PLAYER("pwc1", "left").PASSTO(PLAYER("pwc2", "left")).MINRANGE()).COUNT()
```

It is also possible to restrict the solution space with `ATLEAST(k)` and `ATMOST(k)`. The modifiers will allow the system to return a solution only if the pattern is matched more than `k` times or less than `k` times respectively.

6 Web Application

To facilitate broader access to our research outcome, we have developed a web application as a demonstrative prototype using the SpringBoot[4] Java Framework. The application is hosted online and uses the SoccerNet Game State Reconstruction dataset. A snapshot of this Graphical User Interface (GUI) is displayed on Figure 8.

The first panel on the left allows the user to visualize any of the clips in the dataset. Just below, some example queries are available, and a syntax guide helps the user write their own queries. On the right side, a text-box allows the user to type and search for any queries accepted by the language. The results are displayed just below it. The textual results are available, as well as the extract from the original video during which the solution found is happening.

7 Conclusion & Future Work

This paper introduces TactiCP, a loosely coupled neuro-symbolic approach for retrieving spatio-temporal tactical patterns expressed in a DSL query language from football match videos. The proposed pipeline bridges raw video tracking and high-level tactical analysis by combining neural network–based 2D tracking data, a heuristic module for ball possession extraction, and CP for symbolic pattern retrieval compiled from the DSL queries.

Our core contributions are threefold: (1) an SQL-like DSL designed to express complex, multi-actor football tactics; (2) a compilation methodology that translates these queries into CP models using interval variables and a specialized Substring-Regular constraint; and (3) an open-source web application that validates the system with real-world match data.

Experimental evaluations demonstrate that the system is both expressive, accurately retrieving the frame intervals corresponding to the queries, and computationally viable.

Future work will explore the integration of frequent pattern mining algorithms. Automatically mined patterns could then be formalized, queried, and analyzed in depth using an extended version of our proposed DSL.

[4] `https://spring.io/projects/spring-boot`

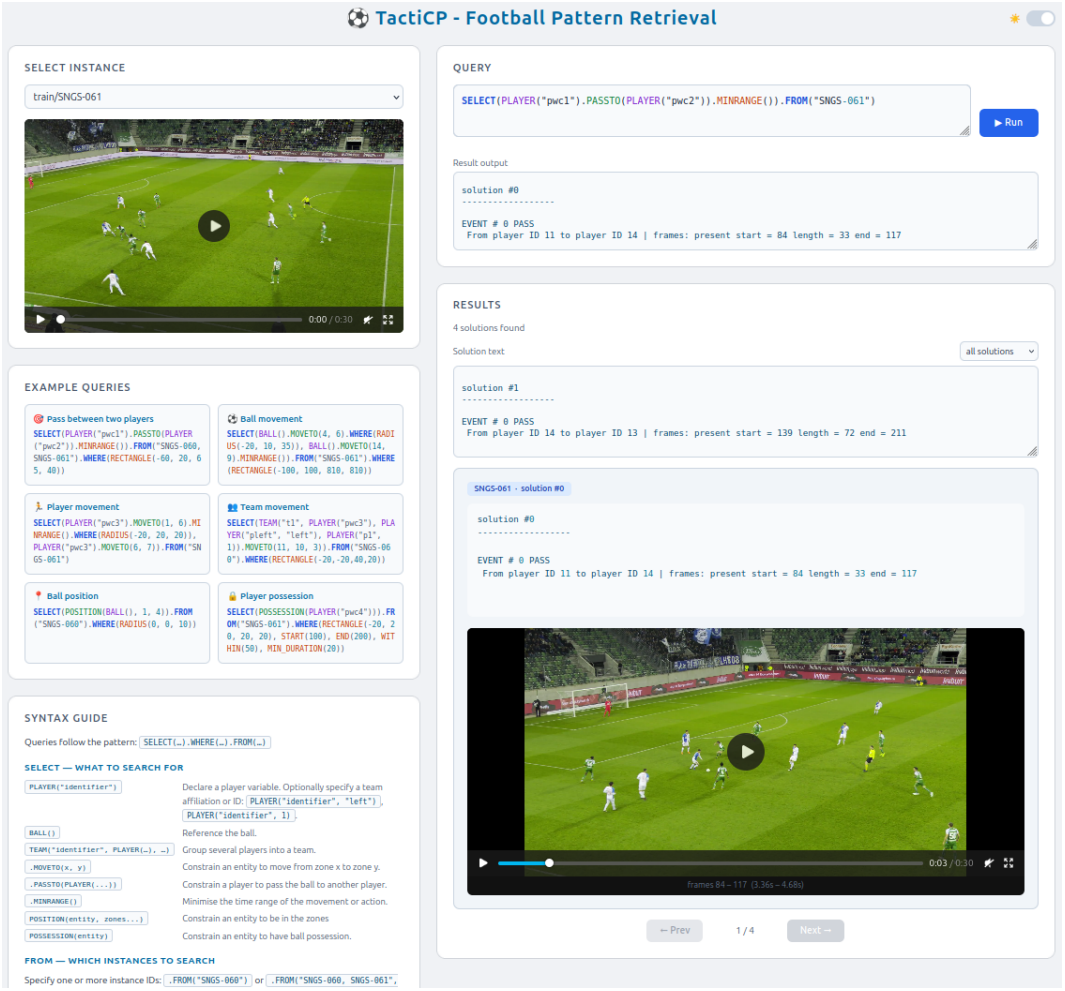


Figure 8 Snapshot of the GUI of the web application, developed to facilitate access to the tool.

References

1 John OR Aoga, Tias Guns, and Pierre Schaus. An efficient algorithm for mining frequent sequence with constraint programming. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 315–330. Springer, 2016. doi:10.1007/978-3-319-46227-1_20.

2 John OR Aoga, Tias Guns, and Pierre Schaus. Mining time-constrained sequential patterns with constraint programming. *Constraints*, 22(4):548–570, 2017. doi:10.1007/s10601-017-9272-3.

3 Relja Arandjelovic, Petr Gronát, Akihiko Torii, Tomás Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. *CoRR*, abs/1511.07247, 2015. URL: http://arxiv.org/abs/1511.07247, arXiv:1511.07247, doi:10.48550/arXiv.1511.07247.

4 Mohamed-Bachir Belaid, Christian Bessiere, and Nadjib Lazaar. Constraint programming for association rules. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 127–135. SIAM, 2019. doi:10.1137/1.9781611975673.15.

5 Quentin Cappart, John OR Aoga, and Pierre Schaus. Episodesupport: A global constraint for mining frequent patterns in a long sequence of events. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 82–99. Springer, 2018. doi:10.1007/978-3-319-93031-2_7.

6 Tom Decroos, Jan Van Haaren, and Jesse Davis. Automatic discovery of tactics in spatio-temporal soccer match data. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, KDD '18, pages 223–232, New York, NY, USA, 2018. ACM. doi:10.1145/3219819.3219832.

7 Adrien Deliège, Anthony Cioppa, Silvio Giancola, Meisam J. Seikavandi, Jacob V. Dueholm, Kamal Nasrollahi, Bernard Ghanem, Thomas B. Moeslund, and Marc Van Droogenbroeck. Soccernet-v2 : A dataset and benchmarks for holistic understanding of broadcast soccer videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021. doi:10.1109/CVPRW53098.2021.00508.

8 Silvio Giancola, Anthony Cioppa, Bernard Ghanem, and Marc Van Droogenbroeck. Deep learning for action spotting in association football videos. *CoRR*, abs/2410.01304:427–459, jul 2024. arXiv:2410.01304, doi:10.48550/arXiv.2410.01304.

9 Silvio Giancola and Bernard Ghanem. Temporally-aware feature pooling for action spotting in soccer broadcasts, 2021. URL: https://arxiv.org/abs/2104.06779, arXiv:2104.06779, doi:10.48550/arXiv.2104.06779.

10 Vladimir Golovkin, Nikolay Nemtsev, Vasyl Shandyba, Oleg Udin, Nikita Kasatkin, Pavel Kononov, Anton Afanasiev, Sergey Ulasen, and Andrei Boiarov. From broadcast to minimap: Achieving state-of-the-art soccernet game state reconstruction, 2025. URL: https://arxiv.org/abs/2504.06357, arXiv:2504.06357, doi:10.48550/arXiv.2504.06357.

11 Tias Guns, Anton Dries, Siegfried Nijssen, Guido Tack, and Luc De Raedt. Miningzinc: A declarative framework for constraint-based mining. *Artificial Intelligence*, 244:6–29, 2017. doi:10.1016/j.artint.2015.09.007.

12 Tias Guns, Siegfried Nijssen, and Luc De Raedt. Itemset mining: A constraint programming perspective. *Artificial Intelligence*, 175(12-13):1951–1983, 2011. doi:10.1016/j.artint.2011.05.002.

13 Laszlo Gyarmati and Xavier Anguera. Automatic extraction of the passing strategies of soccer teams, 2015. URL: https://arxiv.org/abs/1508.02171, arXiv:1508.02171, doi:10.48550/arXiv.1508.02171.

14 K. Kausalya, S. Kanaga Suba Raja, and S. Sudha. An optimized deep learning approach for detection and classification of player actions in football game. *Entertain. Comput.*, 55:101003, 2025. doi:10.1016/j.entcom.2025.101003.

15 Amina Kemmar, Samir Loudni, Yahia Lebbah, Patrice Boizumault, and Thierry Charnois. A global constraint for mining sequential patterns with gap constraint. In *International Confer-*

ence on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems, pages 198–215. Springer, 2016. doi:10.1007/978-3-319-33954-2_15.

16 Philippe Laborie and Jerome Rogerie. Reasoning with conditional time-intervals. In *Proceedings of the 21th International Florida Artificial Intelligence Research Society Conference, FLAIRS-21*, pages 555–560, 01 2008. URL: http://www.aaai.org/Library/FLAIRS/2008/flairs08-126.php.

17 Connor McGillick, Chris Towlson, Steve Barrett, and John Toner. Performance analysis in sport and soccer: Past, present and future–narrative review. *Journal of sports research*, 11(1):19–41, 2024.

18 Benjamin Negrevergne and Tias Guns. Constraint-based sequence mining using constraint programming. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 288–305. Springer, 2015. doi:10.1007/978-3-319-18008-3_20.

19 Yin May Oo, Yewon Hwang, Muhammad Amrulloh Robbani, Vanyi Chao, Ankhzaya Jamsrandorj, Hoang Quoc Nguyen, Kyung-Ryoul Mun, and Jinwook Kim. Broadcast2pitch: Game state reconstruction from unconstrained soccer videos. URL: https://api.semanticscholar.org/CorpusID:286258420.

20 Gilles Pesant. A regular language membership constraint for finite sequences of variables. In Mark Wallace, editor, *Principles and Practice of Constraint Programming – CP 2004*, pages 482–495, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. doi:10.1007/978-3-540-30201-8_36.

21 Pierre Schaus, John OR Aoga, and Tias Guns. Coversize: A global constraint for frequency-based itemset mining. In *International Conference on Principles and Practice of Constraint Programming*, pages 529–546. Springer, 2017. doi:10.1007/978-3-319-66158-2_34.

22 Pierre Schaus, Guillaume Derval, Augustin Delecluse, Laurent Michel, and Pascal Van Hentenryck. Maxicp: A constraint programming solver for scheduling and vehicle routing, 2024. URL: https://github.com/aia-uclouvain/maxicp.

23 João V. B. Soares, Avijit Shah, and Topojoy Biswas. Temporally precise action spotting in soccer videos using dense detection anchors. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2796–2800, 2022. doi:10.1109/ICIP46576.2022.9897256.

24 Vladimir Somers and et al. SoccerNet game state reconstruction: End-to-end athlete tracking and identification on a minimap. In *2024 IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Work. (CVPRW)*, jun 2024. doi:10.1109/CVPRW63382.2024.00334.

25 Manuel Stein, Halldór Janetzko, Tobias Schreck, and Daniel A. Keim. Tackling similarity search for soccer match analysis: Multimodal distance measure and interactive query definition. *IEEE Computer Graphics and Applications*, 39(5):60–71, 2019. doi:10.1109/MCG.2019.2922224.

26 Ferran Vidal-Codina, Nicolas Evans, Bahaeddine El Fakir, and Johsan Billingham. Automatic event detection in football using tracking data, 2022. URL: https://arxiv.org/abs/2202.00804, arXiv:2202.00804, doi:10.48550/arXiv.2202.00804.

27 Xiao Xie, Jiachen Wang, Hongye Liang, Dazhen Deng, Shoubin Cheng, Hui Zhang, Wei Chen, and Yingcai Wu. Passvizor: Toward better understanding of the dynamics of soccer passes. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1322–1331, 2021. doi:10.1109/TVCG.2020.3030359.

28 Hao Xu, Arbind Agrahari Baniya, Sam Well, Mohamed Reda Bouadjenek, Richard Dazeley, and Sunil Aryal. Deep learning for sports video event detection: Tasks, datasets, methods, and challenges, 2025. URL: https://arxiv.org/abs/2505.03991, arXiv:2505.03991, doi:10.48550/arXiv.2505.03991.

29 Mohammed J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1):31–60, Jan 2001. doi:10.1023/A:1007652502315.

30 Boqiang Zhang, Kehan Li, Zesen Cheng, Zhiqiang Hu, Yuqian Yuan, Guanzheng Chen, Sicong Leng, Yuming Jiang, Hang Zhang, Xin Li, Peng Jin, Wenqi Zhang, Fan Wang, Lidong Bing, and Deli Zhao. Videollama 3: Frontier multimodal foundation models for image and video

understanding. *arXiv preprint arXiv:2501.13106*, 2025. URL: https://arxiv.org/abs/2501.13106, doi:10.48550/arXiv.2501.13106.

31 Xin Zhou, Le Kang, Zhiyu Cheng, Bo He, and Jingyu Xin. Feature combination meets attention: Baidu soccer embeddings and transformer based temporal detection, 2021. URL: https://arxiv.org/abs/2106.14447, arXiv:2106.14447, doi:10.48550/arXiv.2106.14447.

A Domain-Specific Language Grammar & Syntax

A.1 Allowed tokens

Identifiers

```
IDENT ::= [a-zA-Z_][a-zA-Z0-9_]*
```

Literals

```
INT     ::= [0-9]+
STRING  ::= '"' .*? '"'
BOOLEAN ::= TRUE | FALSE
```

Reserved Keywords

SELECT, FROM, WHERE, AND, OR, NOT, ALL, TRAIN, TEST, VALID, MAXDURATION, MINDURATION, STARTMIN, STARTMAX, ENDMIN, ENDMAX, RADIUS, RECTANGLE, COUNT, ATMOST, ATLEAST, PLAYER, BALL, TEAM, LEFT, RIGHT, POSSESSION, POSITION, PASSTO, MOVETO.

A.2 Query Structure

```
query ::= SELECT '(' event_select ')' '.' WHERE '(' where_expr ')' '.' FROM '('
    from_expr ')' ('.' search_mode)?
search_mode ::= SEARCH '(' FIRST | ALL | INT ')'
             | COUNT '(' ')'
```

SELECT[5] on events returns a list of frame intervals, where each interval contains the frames at which each event of the select expression starts and ends. By default, the events in the sequence (and thus the corresponding intervals) will necessarily happen in chronological order.

The default mode (if `search_mode` is not present) is `ALL`, which means that all possible solutions will be a part of the output. It is also possible to output the first solution found only, as well as the first `k` solutions. If `COUNT` is present, the engine will output only the number of solutions, but not the actual variables binding.

SELECT clause

```
event_select ::= '(' event_expr (',' event_expr)* ')'
event_expr  ::= base_event ('.' MINRANGE '(' ')')?
```

An `event_select` comprises one or several events ordered chronologically. The optional `MINRANGE` modifier forces each event's interval to be the minimal contiguous interval where the event predicate holds (e.g. for a pass, from the frame the ball leaves the passer's foot to the frame it reaches the receiver). Without `MINRANGE`, the interval for an event is the largest possession-based interval (e.g. from when the passer first gains possession to when the receiver loses it).

[5] In the actual implementation, it is necessary to launch the search by .search() or .searchAndPrint()

```
event_expr ::= event_atom
             | AND '(' event_expr ',' (event_expr)* ')'
             | OR  '(' event_expr ',' (event_expr)* ')'
             | NOT '(' event_expr ')'

event_atom ::= labeled_event
             | atomic_event
             | '(' event_expr ')'
```

As previously stated, selecting an event i will return a time interval $[start_i, end_i]$. Therefore, `AND`$(Event_A, Event_B)$ means that both events hold at the same time. The result is thus the intersection of the interval sets.

`OR`$(Event_A, Event_B)$ means that at least one of the events must hold. We make use of the optionality of interval variables to enforce that at least one interval is present.

`NOT` $Event_A$ returns all time intervals where $Event_A$ does not hold. Formally, the result corresponds to the complement of the intervals of $Event_A$ within the search horizon.

It is important to note that `NOT` is only defined for events that can be evaluated at a single frame. This is the case for atomic events such as `POSSESSION` and `POSITION`. For events spanning multiple frames, negation is not well defined. For instance, the negation of a pass cannot be meaningfully interpreted.

```
atomic_event ::= POSSESSION '(' entity ')'
               | POSITION '(' entity ',' INT ',' (INT)* ')'
```

`POSSESSION`(Team t) means that at least one of the players of the team has the ball, while `POSSESSION`(Player p) means that the player has possession of the ball.

`POSITION` (Team t, int... zones, int k) means that at least k players among the players of the team are in the zones, `POSITION` (Player p, int... zones) or `POSITION` (Ball b, int... zones) simply means that the player or the ball must be inside the defined set of zones.

```
labeled_event ::= player_entity '.' PASSTO '(' player_entity ')'
                | player_entity '.' MOVETO '(' INT, INT ')'
                | ball_entity '.' MOVETO '(' INT, INT ')'
                | team_entity '.' MOVETO '(' INT, INT, INT ')'
```

The `A.PASSTO(B)` event represents a pass from player A to player B.

The `BALL.MOVETO(zone_start, zone_end)` represents the movement of the ball from `zone_start` to `zone_end`. A similar logic applies to player entities.

The `T.MOVETO(zone_start, zone_end, k)` represents the movement of at least k players from the team T from `zone_start` to `zone_end`.

Entities

```
entity ::= player_entity | ball_entity | team_entity

player_entity ::= PLAYER '(' IDENT ')'
                | PLAYER '(' IDENT ',' team_spec ')'
                | PLAYER '(' IDENT ',' INT ')'
                | PLAYER '(' IDENT ',' team_spec ',' INT ')'
ball_entity  ::= BALL '(' ')'

team_entity  ::= TEAM '(' "LEFT" | "RIGHT" ')'
               | TEAM '(' IDENT ',' INT (',' INT)* ')'
               | TEAM '(' IDENT ',' player_entity (',' player_entity)* ')'
```

A `PLAYER` must have a name (its identifier) and may optionally specify a team ("left" or "right") and/or an ID. If an ID is provided, the team attribute is ignored and replaced by the team associated with that ID. A player may also be defined using only an identifier, in which case it acts as a wildcard. The identifier is then treated as unique: if the same identifier appears again, the system assumes it refers to the same player performing the action.

A `TEAM` must have a name (its identifier). If the name is "LEFT" or "RIGHT", all the players of the corresponding team are considered to be a part of the team. Otherwise, the team can also be created with an arbitrary number of players' ID, or any number of player entities.

WHERE clause

```
where_expr ::= spatial_predicate
             | temporal_predicate
             | aggregation
             | '(' where_expr ')'
temporal_predicate ::= MAXDURATION '(' INT ')'
                     | MINDURATION '(' INT ')'
                     | STARTMIN '(' INT')'
                     | STARTMAX '(' INT')'
                     | ENDMIN '(' INT')'
                     | ENDMAX '(' INT ')'
```

The time is always measured in frames (1/25th of a second). `MINDURATION` restricts the minimal duration of the event or sequence of events, while `MAXDURATION` restricts the maximal duration. `STARTMIN`, `STARTMAX`, `ENDMIN`, and `ENDMAX` restrict the minimal or maximal start and end times.

```
spatial_predicate ::= RECTANGLE '(' INT ',' INT ',' INT ',' INT ')'
                    | RADIUS '(' INT ',' INT ',' INT ')'
```

The distance is always measured in meters.

The `RECTANGLE` constrains the pattern in a spatial rectangle. It requires to define the coordinate (x,y) of the upper left corner of the rectangle, as well as the width (x-axis) and the height (y-axis).

The RADIUS constraints the pattern in a spatial circle. It requires to define the coordinate (x,y) of the center of the circle, as well as the radius.

```
aggregation ::= ATMOST '(' INT ')'
              | ATLEAST '(' INT ')'
```

If ATMOST(k) or ATLEAST(k) appears in the WHERE clause, it restricts the number of solutions the system may return. With ATMOST(k), if a video clip contains at most k occurrences of the pattern, these occurrences are returned normally; otherwise, an error is raised. Conversely, with ATLEAST(k), if the clip contains at least k occurrences, the solutions are returned as usual, while an error is raised if fewer occurrences are found.

FROM clause

```
from_expr ::= match_id
            | match_id (',' match_id)*
            | sub-folder
            | "ALL"
sub-folder ::= "TRAIN"
             | "TEST"
             | "VALID"
             | STRING
```

It is possible to restrict the search to only one match by giving its ID (a string representing the match), a subset of matches or the whole dataset. Current sub-folders are the splits "train", "test", and "valid" orignally present in the SoccerNet Game State Reconstruction dataset.

A.3 Labeled Events as Atomic Events

The two atomic events are POSSESSION and POSITION. Every operation can be built upon them.

```
SELECT(A.MOVETO(ZONE_START,ZONE_END)) ===
SELECT((POSITION(A,ZONE_START), POSITION(A,ZONE_END)))
```

A being a player entity or a team entity and ZONE_START and ZONE_END being integers.

```
SELECT(PLAYER_A.PASSTO(PLAYER_B)) ===
SELECT(POSSESSION(PLAYER_A), NOT(POSSESSION(TEAM_ALL)), POSSESSION (PLAYER_B))
```

PLAYER_A and PLAYER_B being player entities and TEAM_ALL being a team entity of which every player except A and B is a part of.

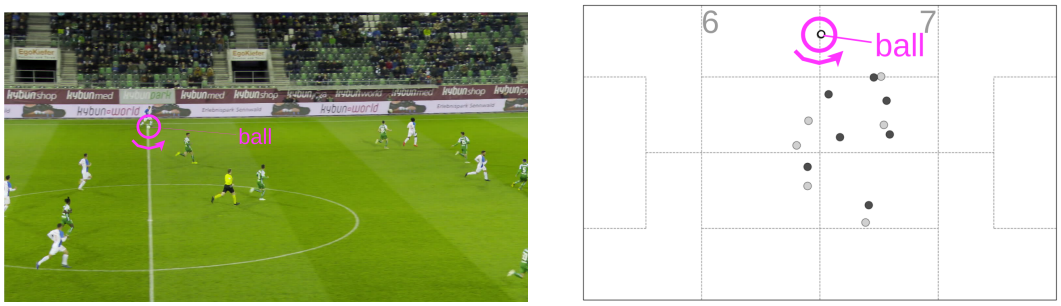
B Other qualitative evaluations

▶ **Example 1** (Basic Movement).

When given the query 1 and an instance of the SoccerNet Game Reconstruction dataset (here "SNGS-061"), the engine returns two solutions during which the ball moves from zone 6 to zone 7. The textual and visual results are displayed in Listing 4 and Figure 9 respectively.

Listing 4 Example 1 textual solution

```
solution #0
------------------
EVENT # 0 BALL MOVES TO
frames: start = 246 length = 2 end = 248
```



(a) Ball moving from zone 6 to 7 (frame 247) **(b)** Corresponding 2D representation

Figure 9 Illustration of a ball movement from zone 6 to 7. The left image shows the original broadcast video, the right image shows the corresponding spatial abstraction used by the system.

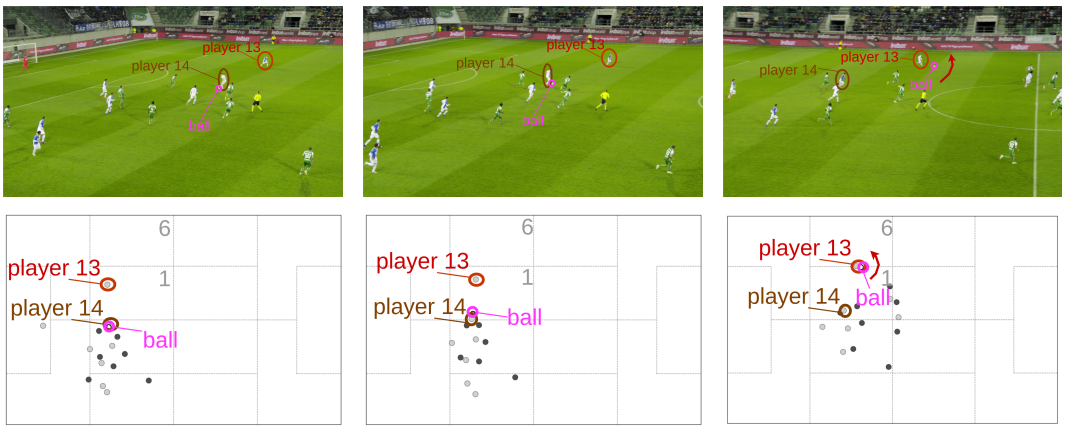
▶ **Example 3** (Temporal & Spatial Constraints).

When given the query 3 and the second instance of the training set (i.e. "SNGS-061"), the engine returns one solution: the sequence of intervals corresponding to when player 14 possesses the ball in a radius of 15 meters centered in (-25,10), then player 13 moves from zone 1 to zone 6 before frame 400. This sequence of events must last less than 100 frames and starts after frame 100. The textual and visual results are displayed in Listing 5 and Figure 10 respectively.

Listing 5 Example 3 textual solution

```
EVENT # 0 HAS BALL
 Player ID 14 | frames: start = 116 length = 23 end = 139

EVENT # 1 MOVES TO
 Player ID 13 | frames: start = 178 length = 2 end = 180
```



(a) Start of possession (frame 116) **(b)** End of possession (frame 139) **(c)** Player 13 moves (frame 179)

Figure 10 Illustration of a sequence of events. The top row shows frames from the original broadcast video, while the bottom row presents the corresponding spatial abstraction used by the system. From left to right, the frames correspond to the start of the possession for player 14, then end of the possession for this player, and the moment player 13 crosses from zone 1 to zone 6.