Recent advances are making possible the use of aggregates in recursive queries thus enabling the declarative expression of classic algorithms and their efficient and scalable implementation. These advances rely on the notion of Pre-Mappability (PreM) of constraints that, along with the seminaive-fixpoint operational semantics, guarantees formal non-monotonic semantics for recursive programs with min and max constraints. In this extended abstract, we introduce basic templates to simplify and automate the task of proving PreM.
Datalog with Extrema

assembly is the maximum number of days required by the basic parts it uses. This can be computed by the following stratified program:

**Example 1** (How many days till all the required parts arrive).

\[
\begin{align*}
\text{deliv}(\text{Part}, \text{Days}) & \leftarrow \text{basic}(\text{Part}, \text{Days}), \text{is\_max}(\text{Part}, \text{Days}). \\
\text{deliv}(\text{Part}, \text{Days}) & \leftarrow \text{deliv}(\text{Sub}, \text{Days}), \text{assbl}(\text{Part}, \text{Sub}). \\
\text{actualDays}(\text{Part}, \text{Days}) & \leftarrow \text{deliv}(\text{Part}, \text{Days}), \text{is\_max}(\text{Part}, \text{Days}).
\end{align*}
\]

But the iterated fixpoint computation of the perfect model of this program can be very inefficient. This problem is solved by transferring \text{is\_max}(\text{Part}, \text{Days}) to \text{deliv}, whereby the rule defining \text{actualDays} now becomes a redundant copy rule. Thus from the previous exo-max version of our program we obtain its endo-max version as follows:

\[
\begin{align*}
\text{deliv}(\text{Part}, \text{Days}) & \leftarrow \text{basic}(\text{Part}, \text{Days}), \text{is\_max}(\text{Part}, \text{Days}). \\
\text{deliv}(\text{Part}, \text{Days}) & \leftarrow \text{deliv}(\text{Sub}, \text{Days}), \text{assbl}(\text{Part}, \text{Sub}), \text{is\_max}(\text{Part}, \text{Days}). \\
\text{actualDays}(\text{Part}, \text{Days}) & \leftarrow \text{deliv}(\text{Part}, \text{Days}).
\end{align*}
\]

The questions we need to answer about our \text{exo2endo} transformation are the following two:

(i) is this a valid optimization inasmuch as the fixpoint computation of the endo-max program delivers the same model as the iterated fixpoint of the original exo-max program (and still allows recursive optimizations such as seminaive-fixpoint and magic-sets)?

(ii) once we re-express \text{is\_max} using negation, does the transformed program has a unique stable model semantics, efficiently computed as described in (i).

The notion of \text{PreM} provides a formal answer to both these questions?

**Definition 2** (The \text{PreM} Property). In a given Datalog program, let \(P\) be the rules defining a (set of mutually) recursive predicate(s). Also let \(T\) be the ICO defined by \(P\). Then, the constraint \(\gamma\) will be said to be \text{PreM} to \(T\) (and to \(P\)) when, for every interpretation \(I\) of \(P\), we have that: \(\gamma(T(I)) = \gamma(T(\gamma(I)))\).

The importance of this property follows from the fact that if \(I = T(I)\) is a fixpoint for \(T\), then we also have that \(\gamma(I) = \gamma(T(I))\), and when \(\gamma\) is \text{PreM} to \(T\) then: \(\gamma(I) = \gamma(T(\gamma(I)))\). Now, let \(T_\gamma\) denote the application of \(T\) followed by \(\gamma\), i.e., \(T_\gamma(I) = \gamma(T(I))\). If \(I\) is a fixpoint for \(T\) and \(I' = \gamma(I)\), then the above equality can be rewritten as: \(I' = \gamma(I) = \gamma(T(\gamma(I))) = T_\gamma(I')\). Thus, when \(\gamma\) is \text{PreM} , the fact that \(I\) is a fixpoint for \(T\) implies that \(I' = \gamma(I)\) is a fixpoint for \(T_\gamma(I)\). In many programs of practical interest, the transfer of constraints under \text{PreM} produces optimized programs for the naive fixpoint computation that are safe and terminating even when the original programs were not. Thus we focus on programs where, for some integer \(n\), \(T_\gamma^n(\emptyset) = T_\gamma^{n+1}(\emptyset)\), i.e., the fixpoint iteration converges after a finite number of steps \(n\). As proven in [3], the fixpoint \(T_\gamma^n(\emptyset)\) so obtained is in fact a minimal fixpoint for \(T_\gamma\), where \(\gamma\) denotes a min or max constraint:

**Theorem 3**. If \(\gamma\) is \text{PreM} to a positive program \(P\) with ICO \(T\) and, for some integer \(n\), \(T_\gamma^n(\emptyset) = T_\gamma^{n+1}(\emptyset)\), then:

(i) \(T_\gamma^n(\emptyset) = T_\gamma^{n+1}(\emptyset)\) is a minimal fixpoint for \(T_\gamma\), and

(ii) \(T_\gamma^n(\emptyset) = \gamma(T_\gamma^{n-1}(\emptyset))\).

Therefore, when the \text{PreM} holds, declarative exo-min (or exo-max) programs are transformed into endo-min (or endo-max) programs having highly optimized operational semantics that computes the perfect model of the former and the unique stable model of the latter.
2 Proving Premappability

The application of a min or max constraint to the ICO of a rule \( r \) can be expressed by the addition of a min or max goal to \( r \), whereby \( \text{PreM} \) holds if this insertion of a new goal does not change the mapping defined by the rule. For the example at hand, we have:

\[
\text{deliv}(\text{Part}, \text{Days}) ← \text{deliv}(\text{Sub}, \text{Days}), \text{is}_{\text{max}}((\text{Sub}), \text{Days}) / \text{assbl}(\text{Part}, \text{Sub}), \text{is}_{\text{max}}((\text{Part}), \text{Days}).
\]

Thus, we must prove that the insertion \( \text{is}_{\text{max}}((\text{Sub}), \text{Days}) / \) does not change the mapping defined by our rule—a property that is guaranteed to hold if can prove that the original mapping already satisfies this constraint. We next define the concept of min- and max-constraints for individual tuples:

- **Definition 4.** We will say that a tuple \( t \in R \) satisfies the min-constraint \( \text{is}_{\text{min}}(X), A \) and write \( X_{\text{min}} \rightarrow A \) when \( R \) contains no tuple having the same \( X \)-value and a smaller \( A \)-value. Symmetrically, we say that the tuple \( t \in R \) satisfies the max-constraint \( \text{is}_{\text{max}}(X), A \) and write \( X_{\text{max}} \rightarrow A \) when \( R \) contains no tuple with the same \( X \)-value and a larger \( A \)-value.

Thus in our example we have \( \text{Part}_{\text{bax}} \rightarrow \text{Days} \) and must prove that \( \text{Sub}_{\text{bax}} \rightarrow \text{Days} \). Toward that goal, we observe that \( X_{\text{bax}} \rightarrow A \) and \( X_{\text{bax}} \rightarrow A \) can be informally viewed as “half functional dependencies (FDs)”, since both must hold before we can conclude that \( X \rightarrow A \).

In fact, although min- and max-constraints on single tuples are much weaker than regular FDs, they preserve some of their important formal properties including those involving multivalued dependencies (MVDs) that result from the natural joins in the recursive rules—e.g. \( \text{Sub} \rightarrow_{\text{Days}} \) and \( \text{Sub} \rightarrow_{\text{Part}} \), in our example.

Therefore, the following properties, proven in [4], hold for tuple for min-constraints, max-constraints, and MVDs, and also illustrate the appeal of the arrow-based notation:

- **Min/Max Augmentation:** If \( X_{\text{min}} \rightarrow A \) and \( Z \subseteq \Omega \), then \( X \cup Z_{\text{min}} \rightarrow A \). If \( X_{\text{max}} \rightarrow A \) and \( Z \subseteq \Omega \), then \( X \cup Z_{\text{max}} \rightarrow A \).
- **MVD Augmentation:** If \( X \rightarrow Y, Z \subseteq \Omega \) and \( Z \subseteq W \), then \( X \cup W \rightarrow Y \cup Z \).
- **Mixed Transitivity:** If \( Y \rightarrow Z \) and \( Z_{\text{min}} \rightarrow A \), with \( A \not\subseteq Z \), then \( Y_{\text{min}} \rightarrow A \). If \( Y \rightarrow Z \) and \( Z_{\text{max}} \rightarrow A \), with \( A \not\subseteq Z \), then \( Y_{\text{max}} \rightarrow A \).

For the example at hand, \( \text{Sub} \rightarrow_{\text{Part}} \) and \( \text{Part}_{\text{bax}} \rightarrow \text{Days} \), by mixed transitivity. Since this constraint holds, the additional goal \( \text{is}_{\text{max}}((\text{Sub}), \text{Days}) / \) enforcing this max constraint does not change it. Q.E.D.

**References**