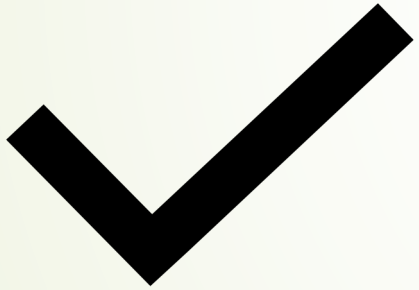
Maximal for Decision Services

Build a Decision Service in 10 Minutes

maximal

- Create a new project
- Define the domain model / schema
- Specify decision logic
  - In Maximal Constraint Language (MCL)
- Start using the API as a decision service



Done!

https://github.com/maximalco/projects

Auto Insurance Eligibility Project

Domain Model

- Insurance Case
- Drivers
- Motor Vehicle Reports (MVR)
- Violations

Decision Logic

- For case to be eligible, all drivers must be eligible.
- For a driver to be eligible:
  - Driver must be over 18 years of age.
  - Driver's MVR must contain
    - 5 or less "Major" violations in total.
    - 3 or less "Major" violations in last 2 years.
  - Driver's MVR must not show current suspension.
- Driver's age = Difference in years between policy effective date and driver's birthdate.

Using the API

- Create a new case with case and driver information.
- Add an MVR report for the driver.
- Query case eligibility.
- Ask for an explanation for eligibility decision.
- Run what-if analyses by modifying values via API.
- Add an additional driver with MVR and check what happens.

Under the Hood

- Create a new project
- Define the domain model / schema
- Specify decision logic
  - In Maximal Constraint Language (MCL)
- Start using the API as a decision service

Auto Insurance Eligibility Project

- Creates a project definition workspace.
- Instantiates a dedicated relational DB schema for the project where all data is stored.

Domain Model

- Maps the specified schema to a relational model.
- Instantiates DB with relational tables, columns, and indices.
  - Tables for cases, drivers, MVRs, violations with relevant columns.
  - Indices and constraints to match relationships as specified in the domain model.

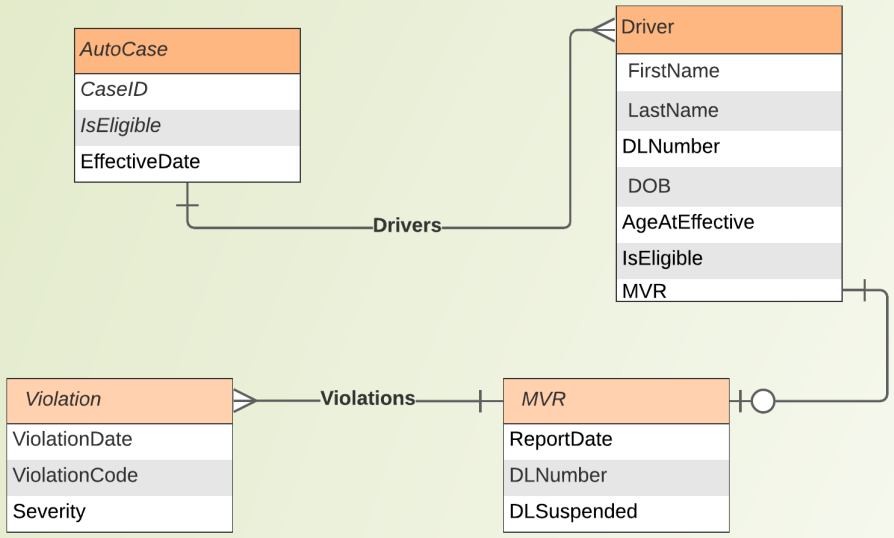
Decision Logic

- Saves the constraint model in the project definition workspace.
- Maintains version information as appropriate.

Using the API

- Persists all submitted case information in the DB.
- Evaluates all data against the decision logic specified and makes inferences.
- Persists any inferred values also in DB.
- Keeps DB state in consistent with given data and decision logic. Always.
  - ”Truth Maintenance”
- Since the case is persisted, it can be invoked anytime in the future and queried for explanations or incrementally modified. The case will adhere to the logic.
  - Maximal is a stateful decision logic engine.

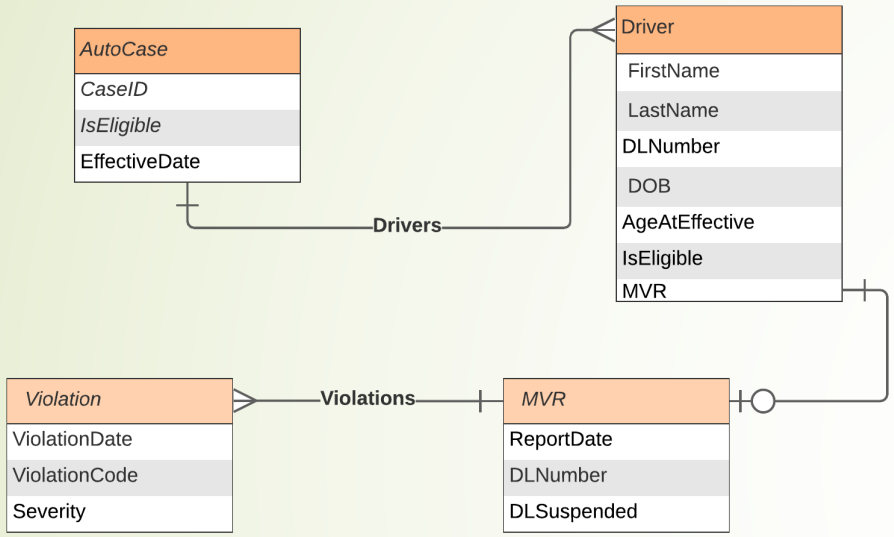
Domain Model



- Objects
  - AutoCase, Driver, MVR, Violation
- Attributes
  - IsEligible, DOB, AgeAtEffective, etc.
- Relations
  - Drivers, Violations

```
<model>
  <object-type name="AutoCase@eg.maximal.co" label="Auto Insurance Case">
    <attribute name="CaseID" label="Id of the case." datatype="text" optional="false"/>
    <attribute name="EffectiveDate" label="Policy effective date" datatype="date"/>
    <attribute name="IsEligible" label="Whether or not case is eligible" datatype="boolean"/>
    <relation name="Drivers" label="Drivers in the case" inverse-object="Driver@eg.maximal.co"
      inverse-attribute="AutoCase" is-containment="true"/>
    <persistence-params table-name="auto_case"/>
  </object-type>

  <object-type name="Driver@eg.maximal.co" label="Driver information.">
    <attribute name="AutoCase" label="The case driver belongs to." datatype="reference"
      ref-type="AutoCase@eg.maximal.co"/>
    <attribute name="FirstName" label="First name of the driver." datatype="text"/>
    <attribute name="LastName" label="Last name of the driver." datatype="text"/>
    <attribute name="DLNumber" label="Driver license number." datatype="text"/>
    <attribute name="DOB" label="Date of birth." datatype="date"/>
    <attribute name="AgeAtEffective" label="Age at policy effective date." datatype="integer"/>
    <attribute name="IsEligible" label="Whether or not driver is eligible" datatype="boolean"/>
    <attribute name="MVR" label="MVR for the driver" datatype="reference" ref-type="MVR@eg.maximal.co"/>
    <persistence-params table-name="auto_driver"/>
  </object-type>

  <object-type name="MVR@eg.maximal.co" label="Motor vehicle report.">
    <attribute name="ReportDate" label="Date of MVR report." datatype="date"/>
    <attribute name="DLSuspended" label="Is driver license suspended?" datatype="boolean"/>
    <relation name="Violations" label="Violations in the MVR" inverse-object="Violation@eg.maximal.co"
      inverse-attribute="MVR" is-containment="true"/>
    <persistence-params table-name="auto_mvr"/>
  </object-type>

  <object-type name="Violation@eg.maximal.co" label="Motor vehicle report violation.">
    <attribute name="MVR" label="The MVR the violation belongs to." datatype="reference"
      ref-type="MVR@eg.maximal.co"/>
    <attribute name="ViolationDate" label="Violation date." datatype="date"/>
    <attribute name="ViolationCode" label="Violation code" datatype="date"/>
    <attribute name="Severity" label="Violation severity" datatype="enumerated"
      enumtype="ViolationSeverityEnum@eg.maximal.co"/>
    <persistence-params table-name="auto_violation"/>
  </object-type>

  <enum-type name="ViolationSeverityEnum@eg.maximal.co" label="Violation Severity Violation">
    <enum-option name="MAJOR" label="Major"/>
    <enum-option name="MINOR" label="Minor"/>
  </enum-type>
</model>
```

Decision Logic



- For case to be eligible, all drivers must be eligible.
- For a driver to be eligible:
  - Driver must be over 18 years of age.
  - Driver's MVR must contain
    - 5 or less "Major" violations in total.
    - 2 or less "Major" violations in the last year.
  - Driver's MVR must not show current suspension.
- Driver's age = Difference in years between policy effective date and driver's birthdate.

```
SET case = AutoCase@eg.maximal.co;
SET driver = case.Drivers;
SET mvr = driver.MVR;

// m_viol is a set of all major violations in an MVR.
SET m_viol = mvr.Violations WHERE m_viol.Severity = MAJOR;

// m_viol_yr is a set of all major violations in an MVR in the last year.
SET m_viol_yr = mvr.Violations WHERE m_viol_yr.Severity = MAJOR AND
   DATEDIFF(case.EffectiveDate, m_viol_yr.ViolationDate, YEARS) < 1;


CONSTRAINT Calc C1 STRICT "Age calculation"
 driver.AgeAtEffective = DATEDIFF(case.EffectiveDate, driver.DOB, YEARS);

CONSTRAINT Eligibility C0 DEFAULT "A case is eligble by default."
   case.IsEligible;

CONSTRAINT Eligibility C1 "A case is eligible only if all drivers are eligible."
   case.IsEligible => driver.IsEligible;

CONSTRAINT Eligibility C2 "Driver must be 18 or over to be eligible."
   driver.IsEligible => (driver.AgeAtEffective >= 18);

CONSTRAINT Eligibility C3 "Driver must have 5 or less major violations."
   driver.IsEligible => SSIZE(m_viol) <= 5;

CONSTRAINT Eligibility C4 "Driver must have 2 or less major violations in the last year."
   driver.IsEligible => SSIZE(m_viol_yr) <= 2;

CONSTRAINT Eligiblity C5 "Driver must have an active license."
   driver.IsEligible => NOT mvr.DLSuspended;
```

Maximal API: Create a New Case


```
[
  {
    "ObjectType": "AutoCase@eg.maximal.co",
    "CaseID": "AC00001",
    "EffectiveDate": "2021-03-27",
    "Drivers": [
      {
        "FirstName": "John",
        "LastName": "Test",
        "DLNumber": "B534521CA",
        "DOB": "2000-01-01",
        "MVR": {
          "ReportDate": "2021-03-27",
          "DLSuspended": false,
          "Violations": [
            {
              "ViolationDate": "2020-04-20",
              "ViolationCode": "V0031X",
              "Severity": "MAJOR"
            },
            {
              "ViolationDate": "2021-01-09",
              "ViolationCode": "V0042C",
              "Severity": "MAJOR"
            },
            {
              "ViolationDate": "2020-08-16",
              "ViolationCode": "V0031X",
              "Severity": "MAJOR"
            }
          ]
        }
      }
    ]
  }
]
```


POST: *{{endpoint}}/scope/new?proj=AutoInsurance*

- A new ”scope” is created with an initial set of objects instantiated.
- Objects are sent in as a JSON body.
- Returns the scope data with its Id that is used for follow up calls.

- Based on the data, case eligibility is derived to be false.
- The constraint “Eligibility.C4” says that for a driver to be eligible, the driver must not have more than 2 major violations. Here there are 3. Hence driver.IsEligible is set to false, which in turn makes case.IsEligible to be false.

```
CONSTRAINT Eligibility C4 "Driver must have 2 or less major violations in the last year."
    driver.IsEligible => SSIZE(m_viol_yr) <= 2;
```

Maximal API: View All Objects

GET: *{{endpoint}}/scope/objects?proj=AutoInsurance&scope=10000001&includeRels=true*

- Returns objects from the requested scope Id.
- *includeRels* specifies whether all relations should be returned.
- All system generated ObjectIDs are also returned with each object.
- External systems can also specify external Ids for objects when they are created.
- Either of these ids can be used to refer to any objects in the system.


```
[
  {
    "ObjectType": "AutoCase@eg.maximal.co",
    "Drivers": [
      {
        "ObjectType": "Driver@eg.maximal.co",
        "ObjectID": 10000001,
        "ObjectTypeID": 10000002,
        "FirstName": "John",
        "DLNumber": "B534521CA",
        "DOB": "Sat Jan 01 00:00:00 UTC 2000",
        "MVR": {
          "ObjectType": "MVR@eg.maximal.co",
          "ReportDate": "Sat Mar 27 00:00:00 UTC 2021",
          "ObjectID": 10000001,
          "Violations": [
            {
              "ViolationDate": "Mon Apr 20 00:00:00 UTC 2020",
              "ObjectType": "Violation@eg.maximal.co",
              "ViolationCode": "V0031X",
              "ObjectID": 10000001,
              "ObjectTypeID": 10000004,
              "MVR": 10000001,
              "Severity": "MAJOR"
            },
            {
              "ViolationDate": "Sat Jan 09 00:00:00 UTC 2021",
              "ObjectType": "Violation@eg.maximal.co",
              "ViolationCode": "V0042C",
              "ObjectID": 10000002,
              "ObjectTypeID": 10000004,
              "MVR": 10000001,
              "Severity": "MAJOR"
            },
            {
              "ViolationDate": "Sun Aug 16 00:00:00 UTC 2020",
              "ObjectType": "Violation@eg.maximal.co",
              "ViolationCode": "V0031X",
              "ObjectID": 10000003,
              "ObjectTypeID": 10000004,
              "MVR": 10000001,
              "Severity": "MAJOR"
            }
          ],
          "ObjectTypeID": 10000003,
          "DLSuspended": false
        },
        "LastName": "Test",
        "AgeAtEffective": 21,
        "IsEligible": false,
        "AutoCase": 10000001
      }
    ],
    "ObjectID": 10000001,
    "ObjectTypeID": 10000001,
    "CaseID": "AC00001",
    "IsEligible": false,
    "EffectiveDate": "Sat Mar 27 00:00:00 UTC 2021"
  }
]
```

Maximal API: Ask for Explanations

GET: *{{endpoint}}/scope/explainVariable?proj=AutoInsurance&scope=10000001&variable=AutoCase@eg.maximal.co:10000001:IsEligible*

- *variable:* object-type:object-id:attribute format
- Returns all constraints and terminal input variables that explain the inference for the decision variable.
- In this case, the constraint Eligibility.C4 and three major violations in the MVR report provide explanation for why IsEligible is false.
- If there are multiple (redundant) explanations, this method will return them all.
- There is another method to fetch the entire explanation path, not just the terminal variables.


```
[
  [
    {
      "variable": "Violation@eg.maximal.co:10000003:Severity",
      "value": "MAJOR",
      "asserted": true
    },
    {
      "variable": "Violation@eg.maximal.co:10000002:Severity",
      "value": "MAJOR",
      "asserted": true
    },
    {
      "variable": "Violation@eg.maximal.co:10000002:ViolationDate",
      "value": "2021-01-09T00:00:00.0Z",
      "asserted": true
    },
    {
      "description": "A case is eligible only if all drivers are eligible.",
      "active": true,
      "constraint": "Eligibility.C1",
      "priority": "NORMAL"
    },
    {
      "description": "Driver must have 2 or less major violations in the last year.",
      "active": true,
      "constraint": "Eligibility.C4",
      "priority": "NORMAL"
    },
    {
      "variable": "Violation@eg.maximal.co:10000003:ViolationDate",
      "value": "2020-08-16T00:00:00.0Z",
      "asserted": true
    },
    {
      "variable": "AutoCase@eg.maximal.co:10000001:EffectiveDate",
      "value": "2021-03-27T00:00:00.0Z",
      "asserted": true
    },
    {
      "variable": "Violation@eg.maximal.co:10000001:Severity",
      "value": "MAJOR",
      "asserted": true
    },
    {
      "variable": "Violation@eg.maximal.co:10000001:ViolationDate",
      "value": "2020-04-20T00:00:00.0Z",
      "asserted": true
    }
  ]
]
```

Maximal API: Add or Modify Objects

```
[

    {
        "ObjectType": "Violation@eg.maximal.co",
        "ObjectID": 10000001,
        "Severity": "MINOR"
    }
]
```

POST: {{endpoint}}/scope/updateObjects?proj=AutoInsurance&scope=10000001

- Here we update one of the violations to Minor from Major.
- This change gets propagated through constraints resulting in inferring case eligibility to be true. That's because the driver no more has more than 2 major incidents in a year.

- References
  - Maximal Data Modeling Concepts
  - Maximal Constraint Language: Concepts and Reference Manual
  - Maximal API Documentation
  - Using Maximal from Your Application

Is Maximal a Rule Engine?

maximal

- Maximal offers everything a rule engine does, but not vice versa.
- Maximal is a **stateful** and **transactional** decision engine.
  - Allows for incremental updates, data streaming, and complex event processing.
  - Readily available decision analytics.
- Maximal employs constraint logic propagation for inferencing.
  - Theoretical sounder with richer modeling constructs
    - Domain reduction, consistency, and truth maintenance.
    - Allows definition of sets (dynamic and conditional) and treats them as first order entities.
      - Set Size, Set Min, Set Max, Set Mean, etc.
- A clean API based service that is quick to build, not an embeddable engine. Usable from any application written in any language.
- Maximal has many additional capabilities to build entire case management / workflow applications.

Analyzing the Model

```
SET case = AutoCase@eg.maximal.co;
SET driver = case.Drivers;
SET mvr = driver.MVR;

// m_viol is a set of all major violations in an MVR.
SET m_viol = mvr.Violations WHERE m_viol.Severity = MAJOR;

// m_viol_yr is a set of all major violations in an MVR in the last year.
SET m_viol_yr = mvr.Violations WHERE m_viol_yr.Severity = MAJOR AND
    DATEDIFF(case.EffectiveDate, m_viol_yr.ViolationDate, YEARS) < 1;


CONSTRAINT Calc C1 STRICT "Age calculation"
  driver.AgeAtEffective = DATEDIFF(case.EffectiveDate, driver.DOB, YEARS);

CONSTRAINT Eligibility C0 DEFAULT "A case is eligble by default."
    case.IsEligible;

CONSTRAINT Eligibility C1 "A case is eligible only if all drivers are eligible."
    case.IsEligible => driver.IsEligible;

CONSTRAINT Eligibility C2 "Driver must be 18 or over to be eligible."
    driver.IsEligible => (driver.AgeAtEffective >= 18);

CONSTRAINT Eligibility C3 "Driver must have 5 or less major violations."
    driver.IsEligible => SSIZE(m_viol) <= 5;

CONSTRAINT Eligibility C4 "Driver must have 2 or less major violations in the last
year."
    driver.IsEligible => SSIZE(m_viol_yr) <= 2;

CONSTRAINT Eligiblity C5 "Driver must have an active license."
    driver.IsEligible => NOT mvr.DLSuspended;
```

- Every constraint in Maximal is a logical statement that must be true. Both below are logical statements, one resembling traditional If…Then, the other simply an equality.
  - driver.IsEligible => (driver.AgeAtEffective >= 18)
  - driver.AgeAtEffective = DATEDIFF(…);
- In Maximal, A => B means:
  - If A is true, then B is true
  - If B is false, then A is false.

  As a result, if driver.AgeAtEffective < 18, driver.IsEligible becomes false.
- Sets are first order constructs in Maximal
  - m_viol and m_viol_yr are both conditional sets.
  - Memberships are managed dynamically and any changes to memberships are propagated uniformly, as we saw from changing the violation severity.
- We made case eligibility to be true, by default. If any other constraint makes case eligibility to be false, then the default is overwritten.

How do you model these in your favorite rules-engine?

A simple model gets very convoluted.

Domain Reduction

Calculating the minimum borrower's FICO score required for a loan transaction:

- FICO score must be above 600
- For non-residents, FICO must be above 650.
- If loan-to-value (LTV) ratio is above 90%, then FICO must be above 680.
- For an investment property in FL, FICO must be over 720.

Domain reduction feature is highly valuable in keeping the model simple and flexible.

How do you model this in a rule engine?

In Maximal, you model is as you see it. No need to aggregate all FICO conditions and write complex logic to figure out the minimum score required.

- loan.FICO > 600
- (borrower.USResidency = NONRESIDENT) => (loan.FICO > 650);
- (loan.LTV > 90) => (loan.FICO > 660);
- (property.State = FL AND loan.purpose = INVEST) => (loan.FICO > 720)

Maximal maintains the value of FICO as a domain and reduces it as the values are propagated though the constraints.

- At the start, FICO = [600, +Infinity).
- If we specify it to be FL investment, then FICO becomes [720, +Infinity]
- If we specify LTV to be 95, then FICO remains [720, +Infinity] as that is a stricter domain than [660, +Infinity].
- If we not remove FL investment, FICO becomes [660, +Infinity].

Domain Reduction: Categorical Variables

There are three loan products: Elite, Premium and Preferred. Elite is most restrictive and Preferred is the least.

Determine the best available product for a given applicant based on the following conditions:

- For Elite eligibility:
  - FICO must be above 740.
  - Annual income must be greater than $250,000.
  - Net worth must be more than $2,000,000.
- For Premium eligibility:
  - FICO must be above 700.
  - Annual income must be greater than $100,000.
  - Net worth must be more than $1,000,000.
- For Preferred eligibility:
  - FICO must be above 640.
  - Annual income must be greater than $50,000.
  - Net worth must be more than $200,000.

Domain reduction is a highly desirable feature for applications such as product selection, product configuration, risk class assignment, etc.

Allows for multiple perspectives to be integrated easily without having to worry about dependencies.

Straight-forward in Maximal. Define loan.products to be a categorical variable with three choices: (ELITE=3, PREMIUM=2, PREFERRED=1). Simply state these constraints anywhere in the model; no need to aggregate them into a table or write if-then-else statements or worry about order of execution. Maximal reduces the choices meeting all constraints automatically through the domain reduction strategy.

- loan.products >= ELITE
  - borrower.FICO > 740
  - borrower.AnnualIncome > 250000
  - borrower.Networth > 2000000;
- loan.products >= PREMIUM
  - borrower.FICO > 700
  - borrower.AnnualIncome > 100000
  - borrower.Networth > 1000000;
- loan.products >= PREFERRED
  - borrower.FICO > 640
  - borrower.AnnualIncome > 50000
  - borrower.Networth > 200000;

➢ At the start with nothing specified, the value of products = [ELITE, PREMIUM, PREFERRED).
➢ Now specify FICO = 720. This will reduce products to [PREMIUM, PREFERRED)
➢ Now specify Annual Income to be 76,000. This will reduce products to [PREFERRED].
➢ And so on.

Lots More…

maximal

For full modeling capabilities of Maximal, please refer to documentation.

- Lookup tables
- Reference tables
- Scoring operations
- Custom functions
- Conditional creation and retraction of objects
- Events / notifications / alerts
- Generative workflows ideally suited for dynamic case management
- Data validations and required field tracking
- Withdrawal and activation of constraints
- Querying for explanations, inference paths, and dependencies.
- …