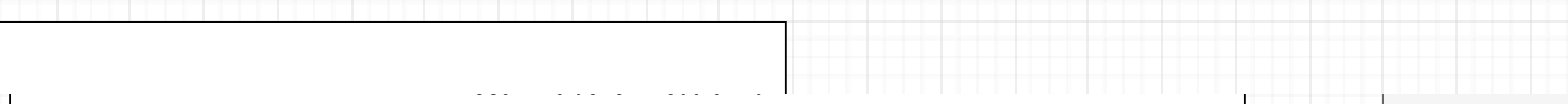
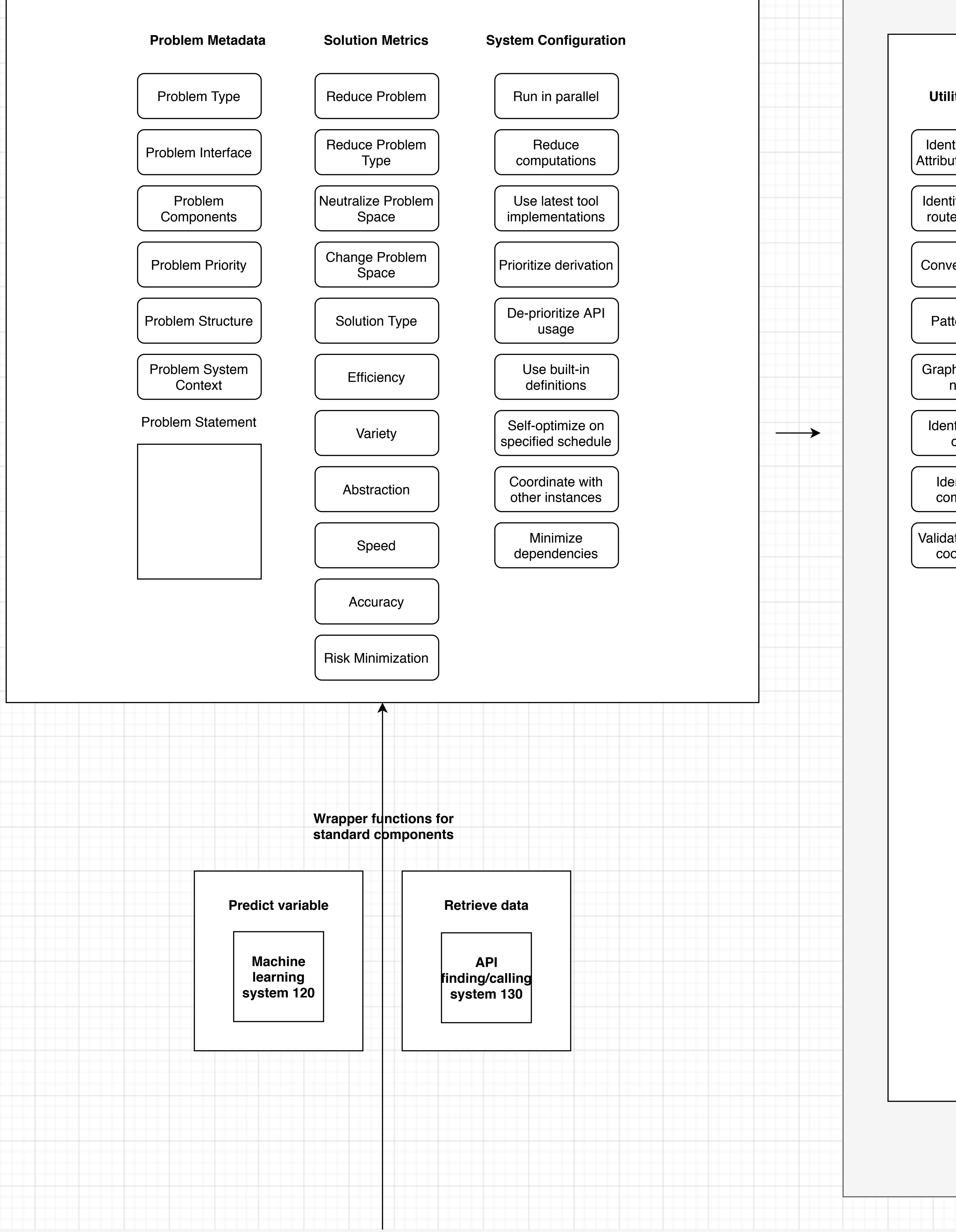
User Interaction Module 110

Figure 1




Problem Metadata
Problem Type
Problem Interface
Problem Components
Problem Priority
Problem Structure
Problem System Context
Problem Statement
Solution Metrics
Reduce Problem
Reduce Problem Type
Neutralize Problem Space
Change Problem Space
Solution Type
Efficiency
Variety
Abstraction
Speed
Accuracy
Risk Minimization
System Configuration
Run in parallel
Reduce computations
Use latest tool implementations
Prioritize derivation
De-prioritize API usage
Use built-in definitions
Self-optimize on specified schedule
Coordinate with other instances
Minimize dependencies
Wrapper functions for standard components
Predict variable
Machine learning system 120
Retrieve data
API finding/calling system 130

Functions

ty Functions

- ify (Object, te, Function)
- fy definition es to object
- ert to format
- ern match
- n system as etwork
- tify system context
- ntify sub- nponents
- te input filter rdination

Graphing Functions

- Select problem dimensions
- Graph problem space
- Identify solution space
- Reduce solution space
- Decompose problem or solution
- Aggregate sub-problems/solutions
- Apply solution to problem
- Convert to interface
- Add/remove dimension
- Compare solutions
- Test solution for metric

Core Functions

- Find
- Build
- Derive
- Change
- Define
- Fit
- Map
- Differentiate
- Identify
- Convert
- Format
- Standardize
- Filter
- Inject
- Compare

Default Interfaces

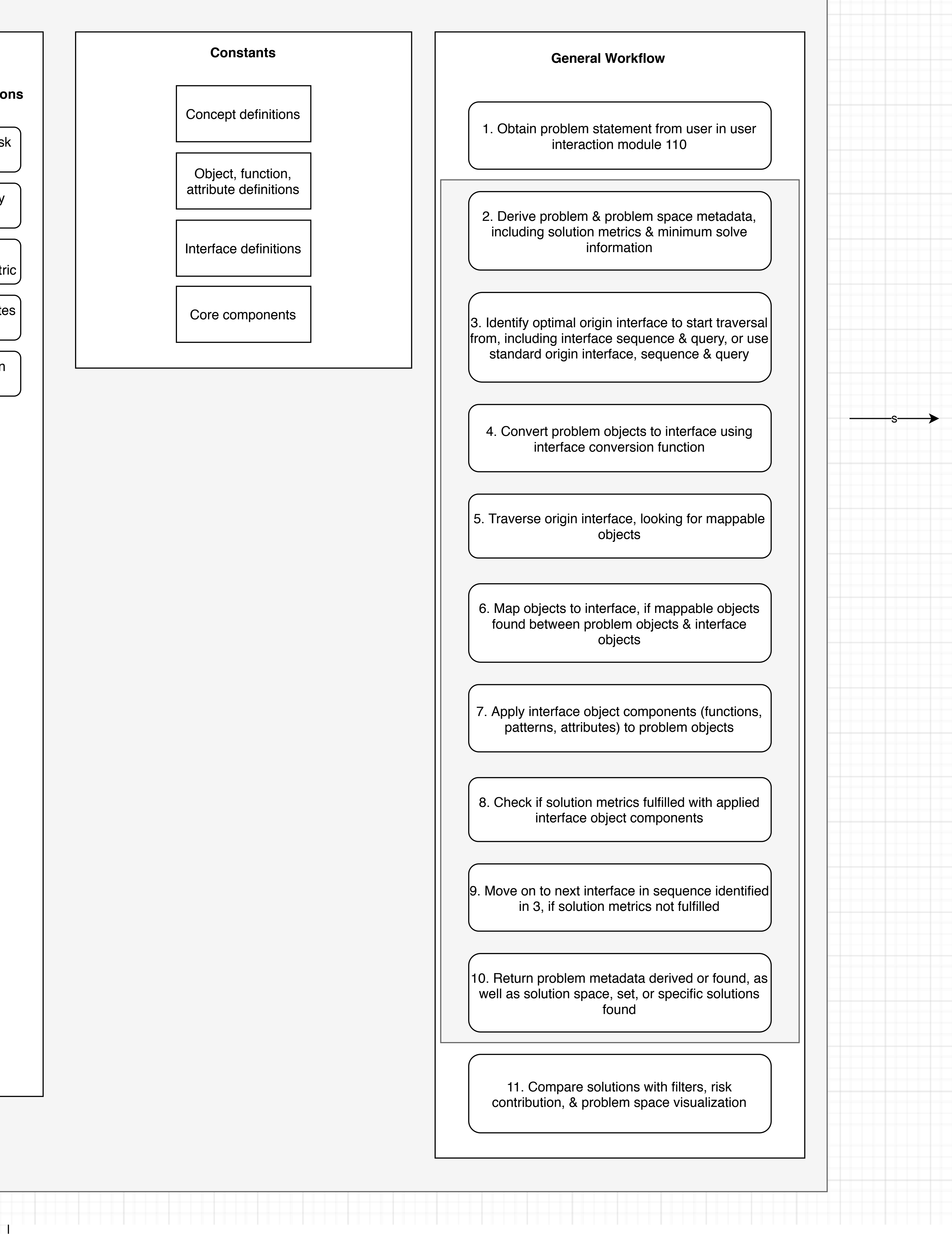
- Concept
- Cause
- Potential/Risk
- Change
- System
- Type
- Attribute
- Function
- Intent
- Information
- Structure
- Math

Interface Functions

- Derive Intent
- Find Important Combinations
- Apply system filters
- Identify unknown objects
- Derive path
- Map systems
- Reduce dependencies
- Close variance injection points
- Enforce rules
- Prioritize concept

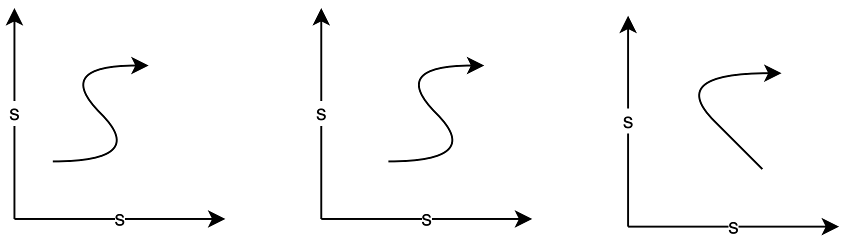
Feedback Functi

- Identify High-Ris Filters
- Compare quer statistics
- Optimize performance met
- Remove duplicat (of a pattern)
- Cache commo traversals


ons
sk
y
tric
tes
n
Constants
Concept definitions
Object, function, attribute definitions
Interface definitions
Core components
General Workflow
1. Obtain problem statement from user in user interaction module 110
2. Derive problem & problem space metadata, including solution metrics & minimum solve information
3. Identify optimal origin interface to start traversal from, including interface sequence & query, or use standard origin interface, sequence & query
4. Convert problem objects to interface using interface conversion function
5. Traverse origin interface, looking for mappable objects
6. Map objects to interface, if mappable objects found between problem objects & interface objects
7. Apply interface object components (functions, patterns, attributes) to problem objects
8. Check if solution metrics fulfilled with applied interface object components
9. Move on to next interface in sequence identified in 3, if solution metrics not fulfilled
10. Return problem metadata derived or found, as well as solution space, set, or specific solutions found
11. Compare solutions with filters, risk contribution, & problem space visualization
s

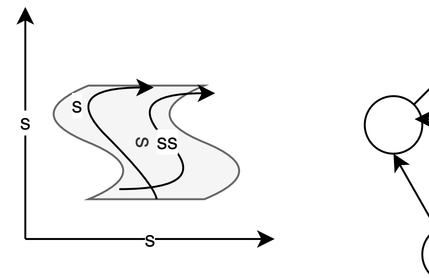
Solution Set

Metric 1 Solution | Metric 2 Solution | Metric 1, 2 Solution



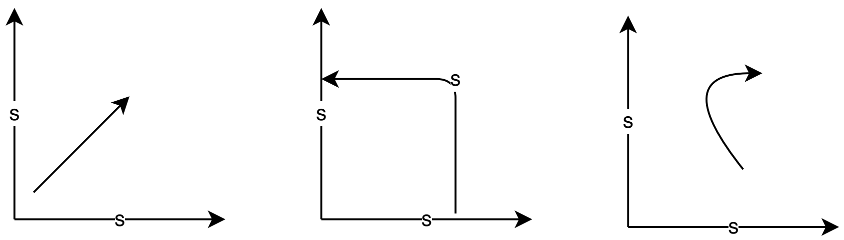
Problem Space Visualization

Solution Space



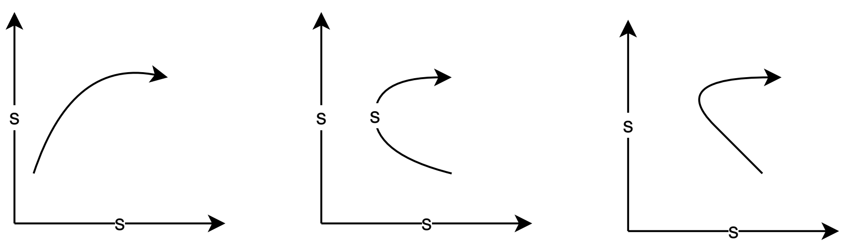
Solution Factors

Shift Factor | **Pivot Factor** | **Rotation Factor**



Alternate Solution Format

Metric 1 Solution | Metric 2 Solution | Metric 1, 2 Solution



Solution Statistics & Metrics

Solution Metrics

Probability of Success:
Pattern ratio used in solution:
Intent stack of solution:
Ratio of insights used to derived:
Attributes:
- most common solution
- most implemented solution
- most retained solution

Solution Statistics

Solution use frequency: 0%
Solution success frequency: 0%

Risk Assessment

Risk added by filters:
Filter 1: 3%
Filter 2: 0%

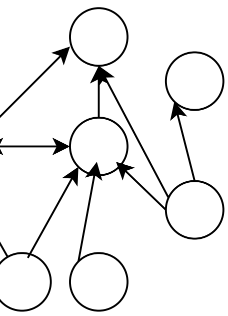
Side effect Risk Assessment:
Side effect 1: 9%
Side effect 2: 3%

Solution Actions

- Show Solution Implentation Steps
- Show Solution Discovery Queries
- Show Solution Impact on Problem Space
- Rate Solution on Success Metrics
- Format Solution (as filters, attributes, etc)
- Show Optimal Solution Discovery Queries
- Show Adjacent Alternative Solutions
- Edit Problem

Component

Problem Space (formatted as a Variable Network)



For a prediction function problem, the solution space is the range of likely prediction functions.

The problem space is the route between independent variables and the dependent variable on a network - it can also be framed as the route between common prediction function terms for a data set like the input data set, and the prediction function. The original problem structure is also depicted as a subset of this problem space visualization.

The solution function can be a route on the problem space if the problem space is formatted as a network, for example.

Solution Components

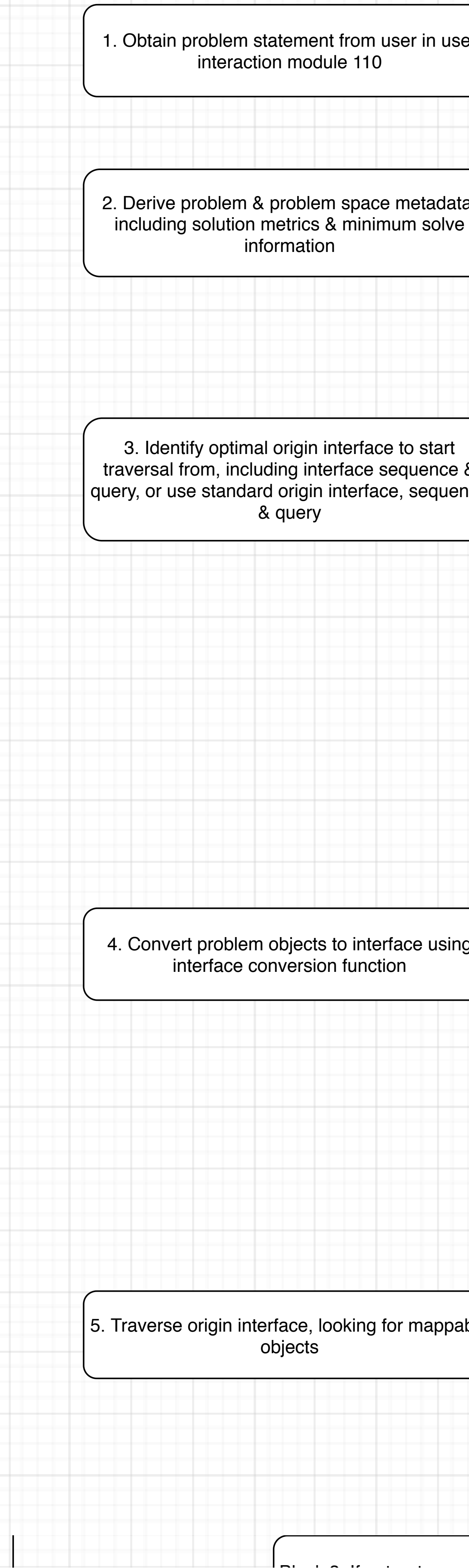
- Pattern A
- Insight B
- System C
- Build Function D
- Filter E
- Network F
- Prediction Function G
- Certainty level H
- Evaluation Metric I
- Test J
- Definition K

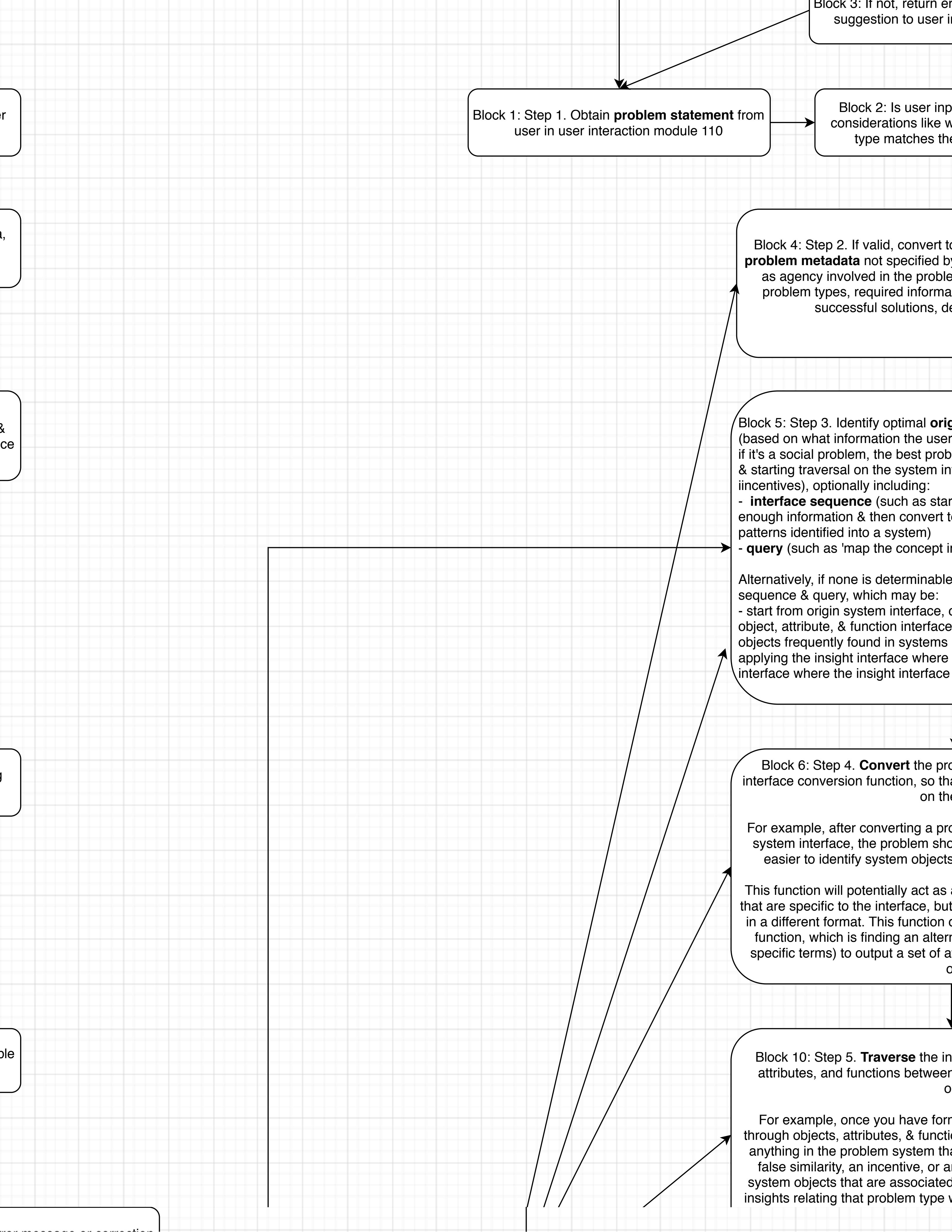
General Workflow

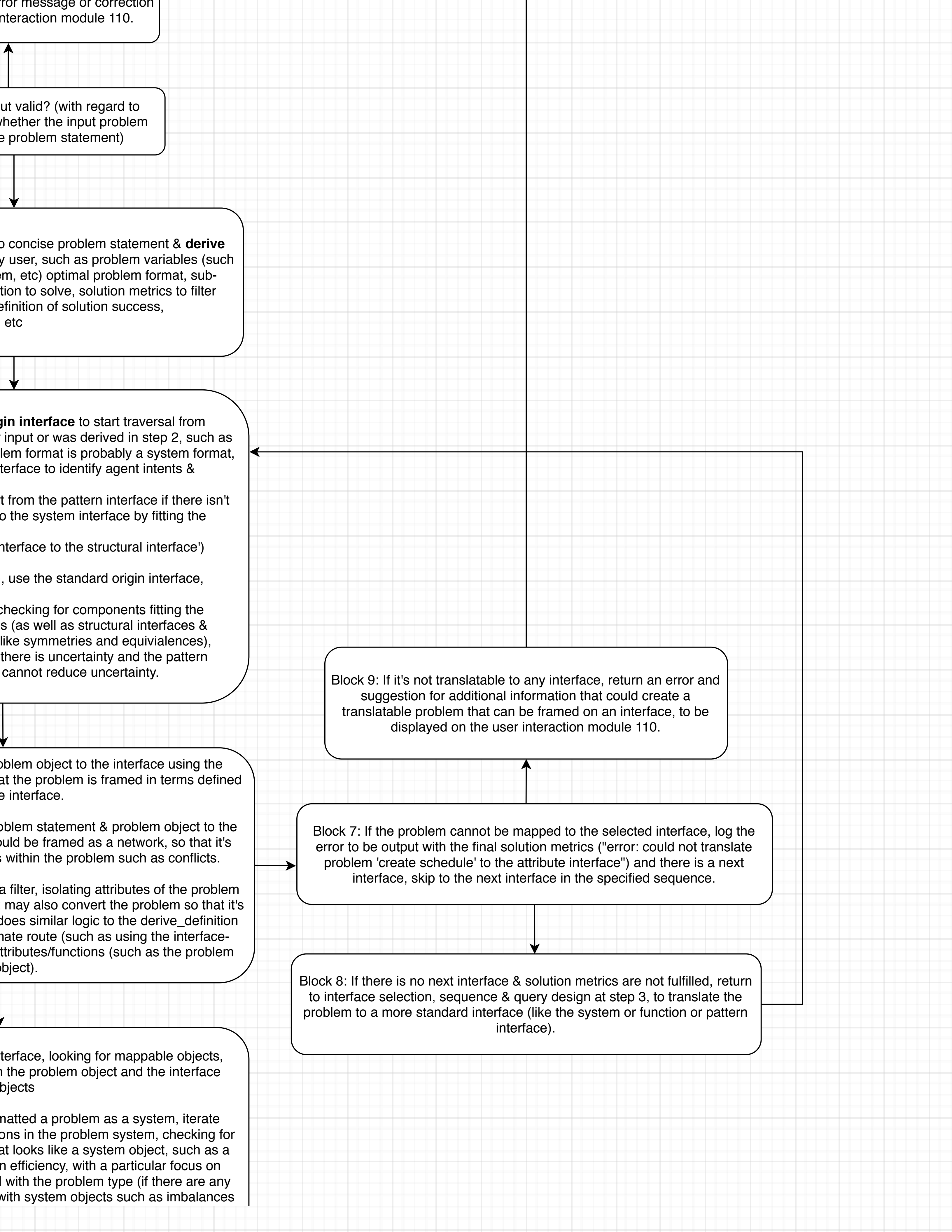
Process 200

Decision Flow

Examples

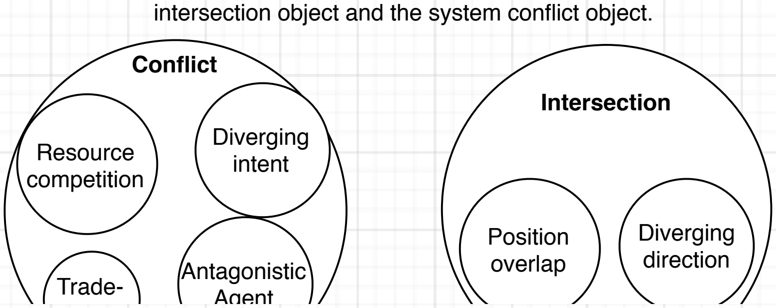

1. Obtain problem statement from user in use
interaction module 110
2. Derive problem & problem space metadata
including solution metrics & minimum solve
information
3. Identify optimal origin interface to start
traversal from, including interface sequence &
query, or use standard origin interface, sequen
& query
4. Convert problem objects to interface using
interface conversion function
5. Traverse origin interface, looking for mappab
objects

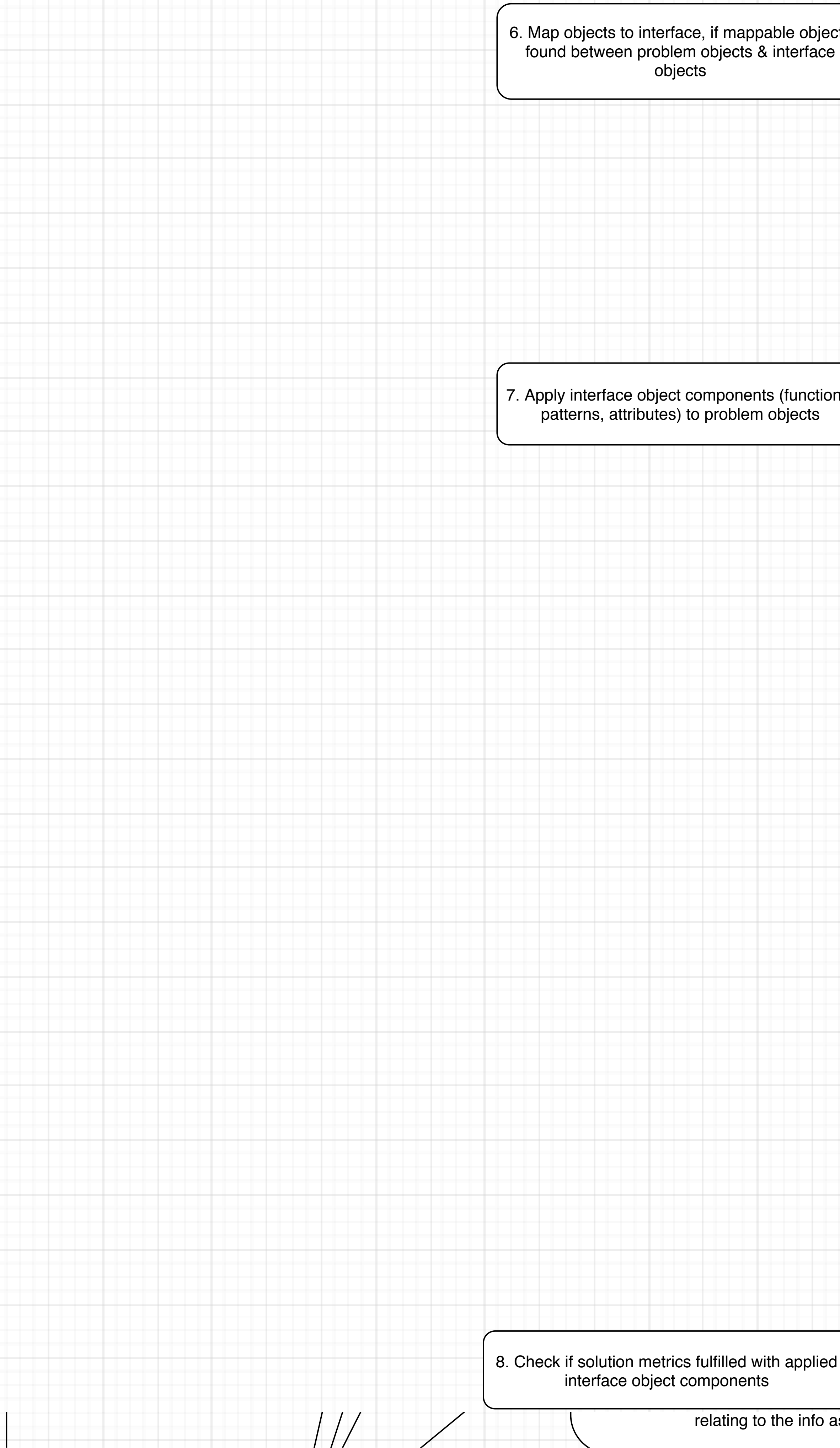

Block 3: If not, return er
suggestion to user i
Block 1: Step 1. Obtain problem statement from user in user interaction module 110
Block 2: Is user inp
considerations like w
type matches the
Block 4: Step 2. If valid, convert to
problem metadata not specified by
as agency involved in the proble
problem types, required informa
successful solutions, de
Block 5: Step 3. Identify optimal orig
(based on what information the user
if it's a social problem, the best prob
& starting traversal on the system in
iincentives), optionally including:
- interface sequence (such as star
enough information & then convert to
patterns identified into a system)
- query (such as 'map the concept i
Alternatively, if none is determinable
sequence & query, which may be:
- start from origin system interface, c
object, attribute, & function interface
objects frequently found in systems
applying the insight interface where
interface where the insight interface
Block 6: Step 4. Convert the pro
interface conversion function, so tha
on the
For example, after converting a pro
system interface, the problem sho
easier to identify system objects
This function will potentially act as a
that are specific to the interface, but
in a different format. This function c
function, which is finding an altern
specific terms) to output a set of a
Block 10: Step 5. Traverse the in
attributes, and functions between
For example, once you have form
through objects, attributes, & functio
anything in the problem system tha
false similarity, an incentive, or a
system objects that are associated
insights relating that problem type v



rror message or correction
nteraction module 110.
ut valid? (with regard to
whether the input problem
e problem statement)
o concise problem statement & derive
y user, such as problem variables (such
em, etc) optimal problem format, sub-
tion to solve, solution metrics to filter
efinition of solution success,
etc
gin interface to start traversal from
input or was derived in step 2, such as
lem format is probably a system format,
terface to identify agent intents &
t from the pattern interface if there isn't
o the system interface by fitting the
nterface to the structural interface')
, use the standard origin interface,
checking for components fitting the
s (as well as structural interfaces &
like symmetries and equivialences),
there is uncertainty and the pattern
cannot reduce uncertainty.
Block 9: If it's not translatable to any interface, return an error and suggestion for additional information that could create a translatable problem that can be framed on an interface, to be displayed on the user interaction module 110.
oblem object to the interface using the
at the problem is framed in terms defined
e interface.
oblem statement & problem object to the
uld be framed as a network, so that it's
s within the problem such as conflicts.
a filter, isolating attributes of the problem
may also convert the problem so that it's
does similar logic to the derive_definition
nate route (such as using the interface-
ttributes/functions (such as the problem
bject).
Block 7: If the problem cannot be mapped to the selected interface, log the error to be output with the final solution metrics ("error: could not translate problem 'create schedule' to the attribute interface") and there is a next interface, skip to the next interface in the specified sequence.
Block 8: If there is no next interface & solution metrics are not fulfilled, return to interface selection, sequence & query design at step 3, to translate the problem to a more standard interface (like the system or function or pattern interface).
terface, looking for mappable objects,
n the problem object and the interface
bjects
matted a problem as a system, iterate
ons in the problem system, checking for
at looks like a system object, such as a
n efficiency, with a particular focus on
d with the problem type (if there are any
with system objects such as imbalances

Example of determining possible match between the problem system intersection object and the system conflict object.





6. Map objects to interface, if mappable object
found between problem objects & interface
objects
7. Apply interface object components (function
patterns, attributes) to problem objects
8. Check if solution metrics fulfilled with applied
interface object components
relating to the info a

ts

ns,

Block 24: Determine which step is necessary to execute change made to problem space visualization in step 11.

Block 11: Step 6. If mappable objects
interface objects, **map** the problem t
object & a degree of certainty i
attributes/functions/ob

For example, if while iterating thro
possible similarity between a problem
conflict object (a similarity in shape or
conflict' label to the problem

Block 12: Step 7. Apply interface ob
attributes) to p

For example, if you find a possible
system in step 6, apply the system c
the problem system

This means if the system conflict o
definition like 'diverging intents' or
'antagonistic agent', apply those to the
they fit in the problem system intersec
intersection point involving differen
'resource competition' and 'divergin
conflict object definition, but may not c
intersection may just be an incidental
for that position, and may allow multip
and the directions may not indicate
both directions). Then the program wo
a query to the insight interface, apply
the problem system once the syste

Block 13: Step 8. Check if solution
object c

symmetry problem type).

s are found between problem objects & to the interface by labeling the problem n the identification, as well as the jects found to be similar.

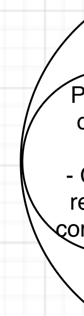
ough the problem system, you find a n system intersection object & a system other attribute value), apply the 'system m system intersection object.

ject components (functions, patterns, problem objects.

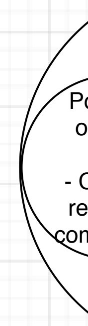
system conflict object in the problem onflict objects, functions, & attributes to m intersection object.

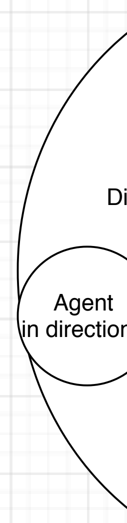
bject has an associated function in its 'trade-off' or 'resource competition' or e problem system intersection and see if ction. An intersection is an overlap at the nt directions, so it's likely to match the ng intents' components of the system depending on the problem definition (the routing object, rather than a competition ble objects occupying the same position, different intents if similar objects are in ould follow this analysis for example with ing any insight objects matched there to em objects were identified & applied.

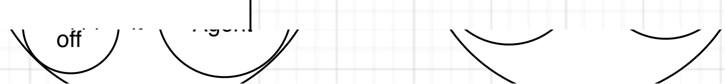
metrics are fulfilled with applied interface components.



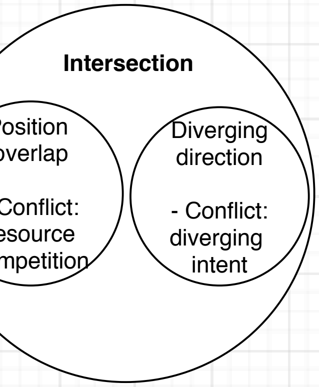
Example of appying
object, where potenti
objects and probable a



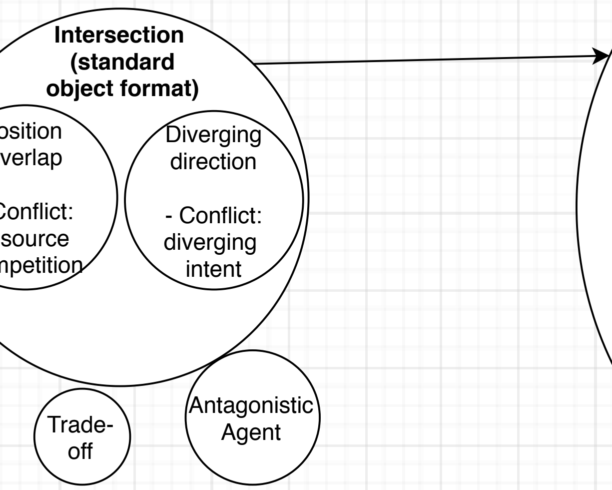


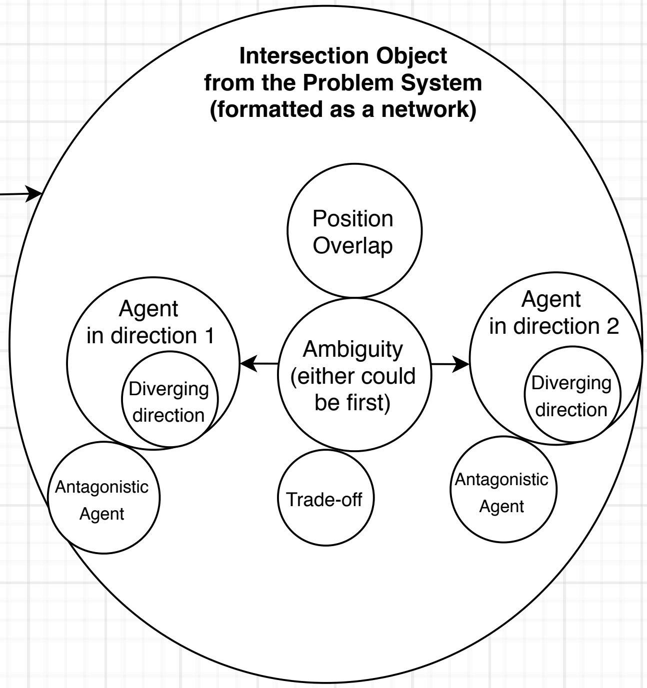


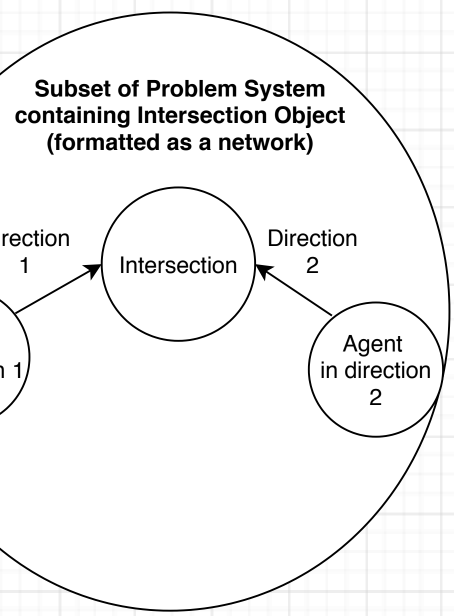
Example of labeling a problem system object like an intersection with the possible matching object in the system interface (and a level of certainty added by each matching attribute/function) which is a conflict object, based on certain conflict attributes from its definition like diverging intents and resource competition.



g interface object components to the problem
ial attributes/functions are included outside of
attributes/functions are contained in the objects





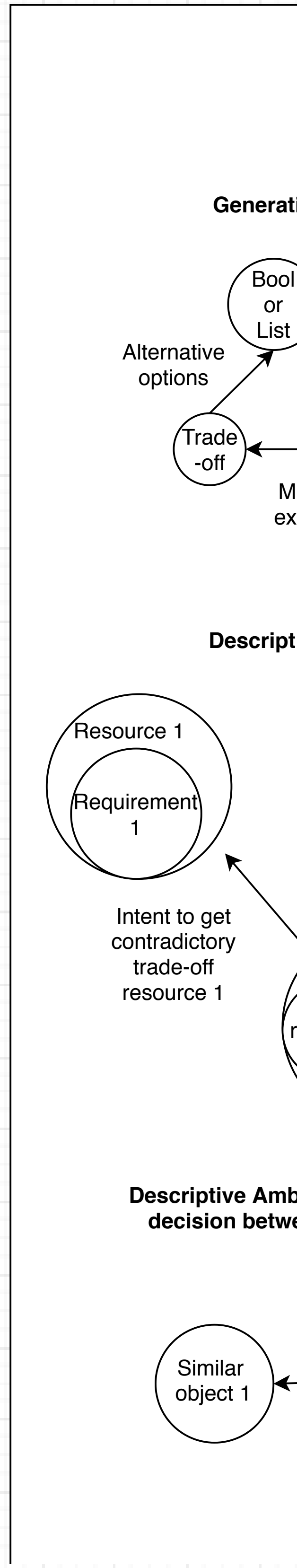


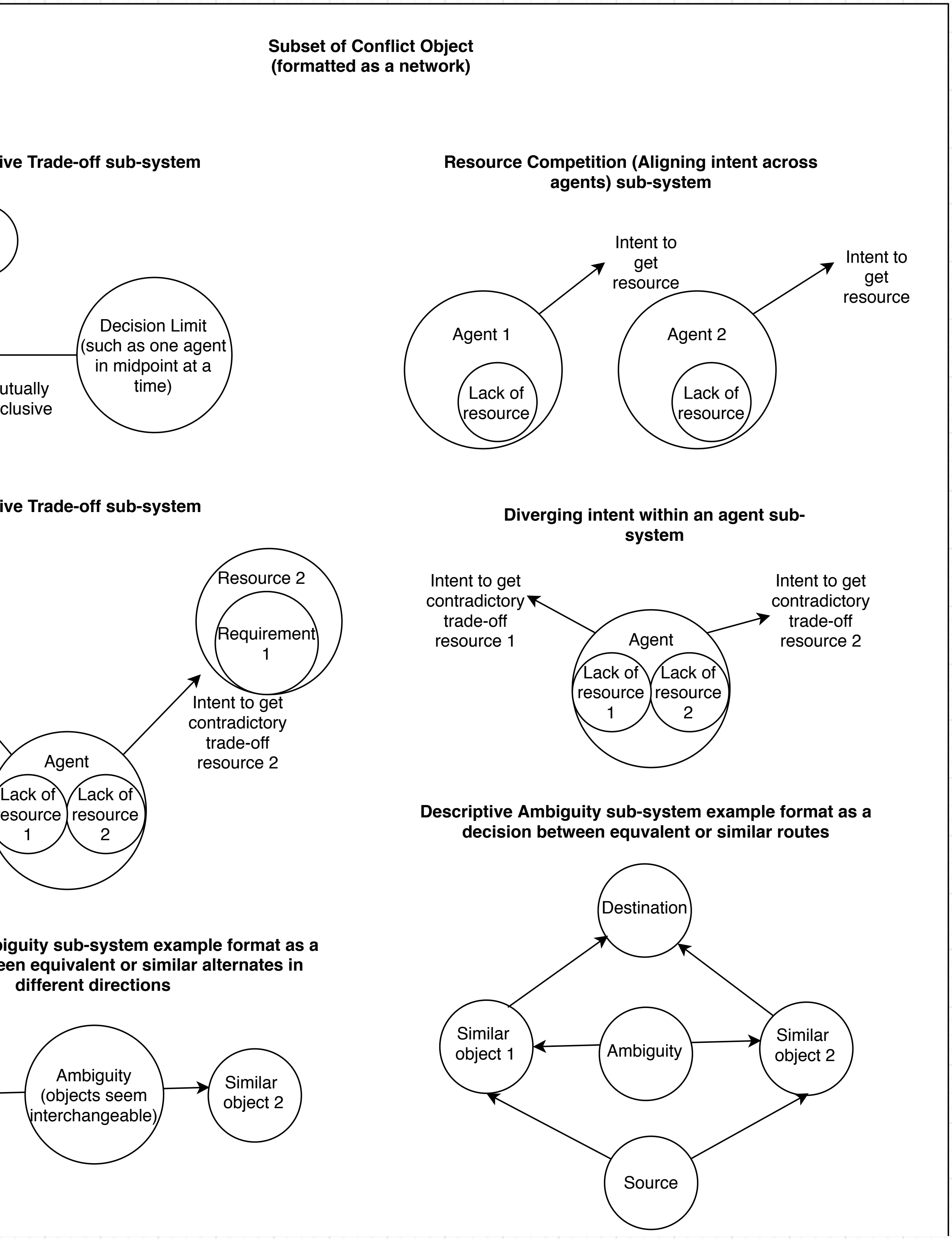
Now the intersection is formatted as a network, and the system objects like conflict (and its sub-components, patterns, objects, etc) have been applied to the intersection network. In the network format, position & other types of connections have semantic value.

Now it's clearer that the intersection has an ambiguity in the position sequence attribute (a variant of the position overlap where only one agent can possess the resource at a given time), creating a possible conflict (determined by which agent arrives in the position first and which agent gets the position resource first).

The diverging direction attribute inherent to the intersection has not been converted to a diverging intent, but it could be if the different directions indicate different intents, and if that difference is relevant to the resolution of the conflict about which agent is allowed in the position first.

The mapping function has also identified a possible trade-off in the ambiguity, indicating that only one agent can occupy the position at a given time, so only one agent can go first (a scarce resource of occupation sequence or saved time that may be causative in the system, especially if the agent changes the intersection or removes some of its value by occupying it first).





Subset of Conflict Object
(formatted as a network)
ive Trade-off sub-system
utually
clusive
Decision Limit
(such as one agent
in midpoint at a
time)
Resource Competition (Aligning intent across
agents) sub-system
Intent to
get
resource
Intent to
get
resource
Agent 1
Lack of
resource
Agent 2
Lack of
resource
ive Trade-off sub-system
Resource 2
Requirement
1
Intent to get
contradictory
trade-off
resource 2
Agent
Lack of
esource
1
Lack of
resource
2
Diverging intent within an agent sub-
system
Intent to get
contradictory
trade-off
resource 1
Intent to get
contradictory
trade-off
resource 2
Agent
Lack of
resource
1
Lack of
resource
2
biguity sub-system example format as a
een equivalent or similar alternates in
different directions
Ambiguity
(objects seem
interchangeable)
Similar
object 2
Descriptive Ambiguity sub-system example format as a
decision between equvalent or similar routes
Destination
Similar
object 1
Ambiguity
Similar
object 2
Source

9. Move on to next interface in sequence identified in 3, if solution metrics not fulfilled

10. Return problem metadata derived or found, as well as solution space, set, or specific solutions found

For example, once you format the
objects in the problem & apply their

Block 15: If there is no next interface & solution metrics are not fulfilled, return to interface selection, sequence & query design at step 3, to translate the problem to a more standard interface (like the system or function or pattern interface).

Block 14: Step 9. If solution metrics not fulfilled, move on to next interface in sequence identified in 3 if there is one, and iterate through process detailed in 3 - 7 to adjust interface sequence or query, convert to the interface, find similar objects between the problem & interface, then apply interface objects & check if the solution metrics are fulfilled by that application.

Block 16: If there is no next interface in the sequence/query and the standard interface origin/sequence/query have been applied already, return information generated/derived/found, including the processes tried & results, to the user interaction module 110.

problem as a system and identify system
r objects, functions, & attributes, is it clear
s other functionality need to be applied?

how to solve the problem or does

Once you identify all the conflicts &
insights toward the minimum inform
is there a clear route or transforma
d

For example, with the intersection o
application which agent should go
possible in the intersection that will
does other functionality need to
systems such as finding substitutes
invalidate a conflict of the

Block 17: Step 10. Return proble
solution space, solution set, or s
ranked or as comparable alterna

For a prediction problem, this me
specific optimal function (solution
with varying bias or other error m
range of functions (solution spac

The program output may include
- input filters
- risk contributed by input filters
- risk contributed by traversals (u
contributes risk)
- solution(s) and/or solution spac
- solution implementation steps
- solution components
- visualization of solution impact
- set of queries used to generate
- methods to generate optimizati
will store for any future users wit
- other soluiton information, like s
ratio of patterns to insights used

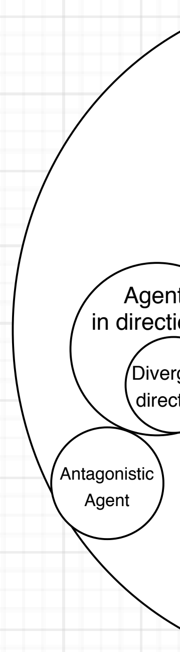
- any non-fatal errors encountere
or components, or patterns/pred

- any problem space information
identified possible/probable insig
causes, etc.

s other functionality need to be applied?

incentives in the problem system & apply
nation to solve and/or the solution metrics,
ation that removes the problem as it was
efined?

object, is it clear from the system interface
first, or whether there is an optimization
invalidate the conflict of who goes first, or
be applied, such as other conflict sub-
s of a resource (like an alternate route) to
resource competition sub-type?



Th
solut
goe

If th

If the

If the i

em metadata derived or found, as well as
pecific optimal solutions found, either
atives

eans returning the function definition if a
n) was found, or the function & variants
metrics optimized (solution set), or a
ce).

e:

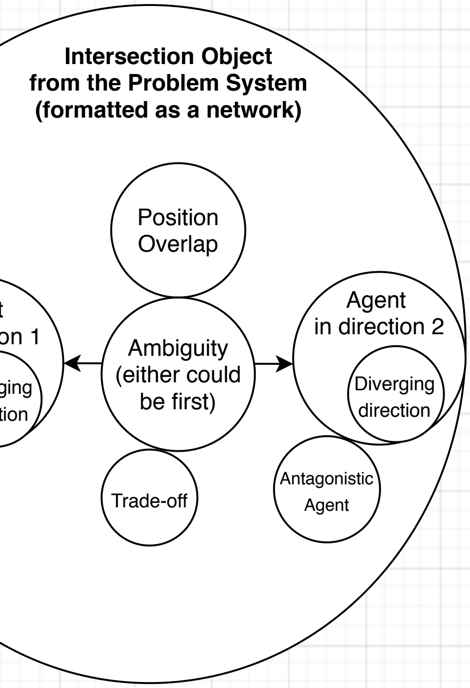
using a pattern instead of an insight

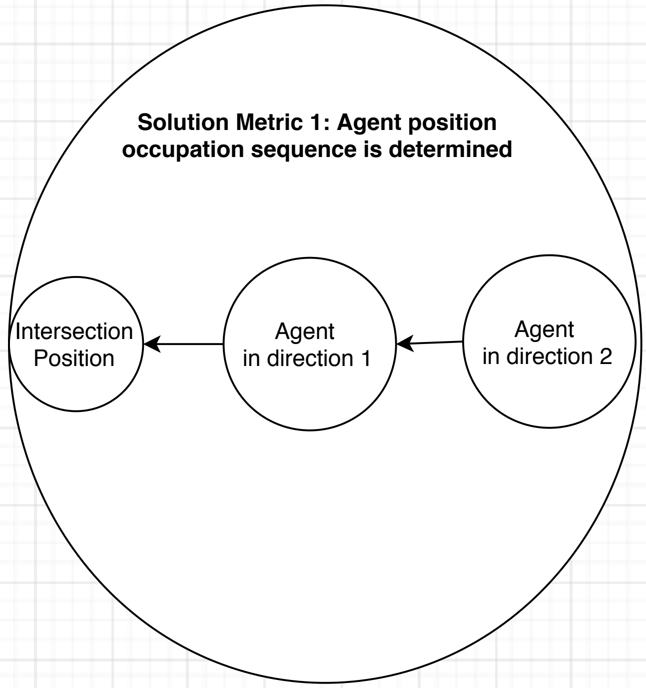
ce

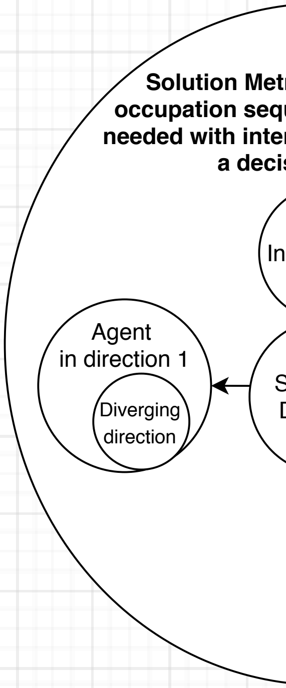
on problem space
e/find/derive solutions
ons of those queries which the system
h a similar problem
solution statistics, success probability,
in the solution, etc.

ed, such as missing optional information
ictions made in the absence of clarity

derived during the traversal, such as
ghts, questions, strategies, patterns,





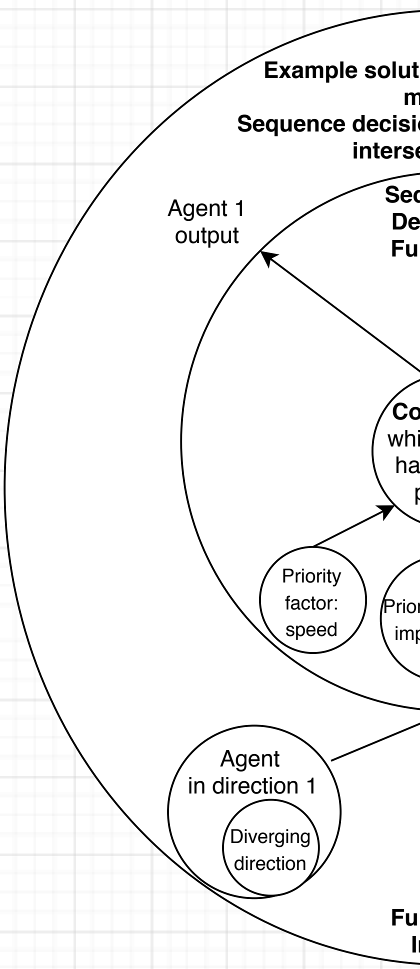


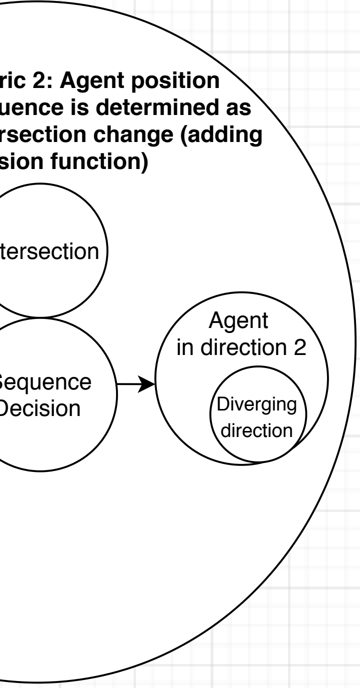
his step identifies whether the output of step 7 creates information that is easily transformed into the
tion metric, given the relevant objects/attributes/functions of the solution metric. Is it clear which agent
es first, or whether the intersection can be changed in a way that determines which agent goes first?

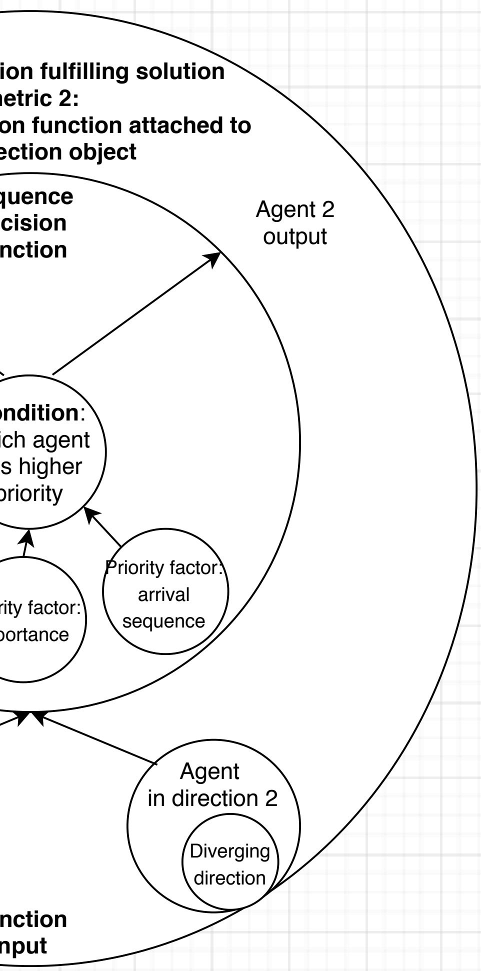
he solution metric 1 is fulfilled, the agents have no antagonistic agent attribute & there is no trade-off
because no variance from a decision is allowed at the intersection.

solution metric 2 is fulfilled, the intersection loses its position overlap attribute & the diverging direction
attribute doesn't matter anymore, but it does have a decision function at the intersection.

intersection object with the system interface is applied can be easily transformed into having one of the
solution metrics fulfilled, that transformation can be considered a possible solution.




ric 2: Agent position
uence is determined as
rsection change (adding
sion function)
tersection
Sequence
Decision
Agent
in direction 2
Diverging
direction


ion fulfilling solution
etric 2:
on function attached to
ection object
quence
cision
nction
Agent 2
output
ondition:
ich agent
s higher
priority
rity factor:
portance
Priority factor:
arrival
sequence
Agent
in direction 2
Diverging
direction
nction
nput

11. Compare solutions with filters, risk contribution, & problem space visualization

Block 18: Step 11. Compare & e
space, describe solution steps &
optimize the traversal & program
interface 110.

This involves listing some proce
interim information derived durin
risk contributed by processing.

It may also involve calculating so
information in the database such
or feedback on solutions entered
output in the user interaction mo
report & accessible with the use

This step is also where the user
component & examine the impa
embedded object graphs in the
change problem objects, & adju
intent,, which may trigger an exe
conversion & traversal process

Users can also download solutio
traversal (skipping unnecessary
that generated the solution, revi
pattern or other risky object dep
generation process, & execute

Block 22: If a re-calculation is necessary, determine which step contains the functionality for that re-calculation, and return flow to that step, to create a new version of the solution output sent to the user interaction module 110.

Block 21: Step 12: Is a re-calculation necessary or is the impact of that change already determined by output?

Block 23: If a re-calculation isn't necessary, adjust the problem space visualization using visualization logic & output information.

Block 19:

valuate solutions, visualize problem
& traversals to generate them, and
n execution, in the user interaction

sses & components used as well as
g the traversal(s), and errors found or

ome solution statistics given stored
n as information about previous queries
d by user on returning to the solution
odule (stored as its own problem output
r interaction module).

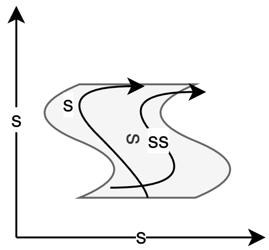
can edit the problem space visualization
ct of other solutions, drill down into
problem space, move or otherwise
st displayed dimensions of the space like
ecution of the problem definition, interface
depending on the edits made.

on steps, optimize the system or the
nodes & so on), examine the queries
ew the risk contributed by each filter or
ended on by the solution or solution
other actions on the output information.

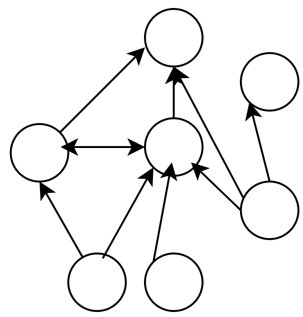
Block 19:
Solution 150

Problem Space Visualization (component of the response to the user interaction module 110 once formatted)

Solution Space



Problem Space (formatted as a Variable Network)



Problem S
Dimensio

- Importance
- Complexity
- **Relevance**
- **Dependency**

Problem Space

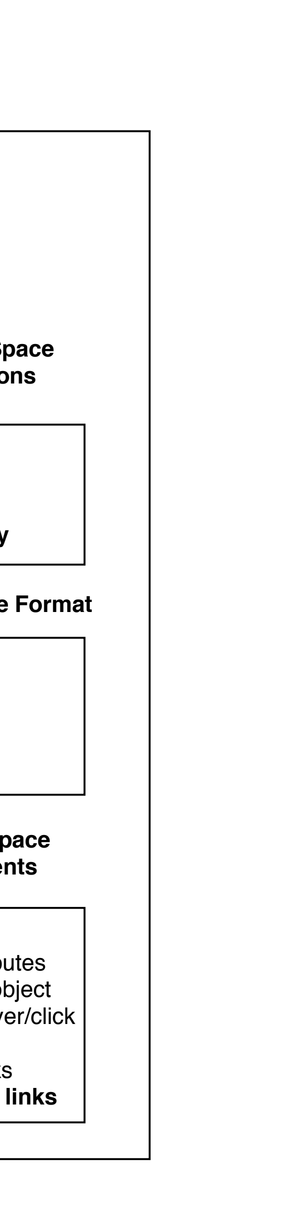
- **N**etwork
- Attribute
- Function
- Filter
- Math

Problem Sp
Compone

- **Labels**
- Unique attrib
- Embedded o
graphs on hov
- Intent
- Function link
- Connection

Block 20: User edits problem space visualization.

This step can involve user edits to the problem space visualization component of the user interf
including edits like changing the position or other attributes of problem objects & their attributes/f
different solutions in the solution set, changing the dimensions of the problem space. When th
problem space visualization, the changes are sent to step 2 or later (depending on whether ad
problem definition or conversion to the interface needs to be done & so on), where the calculati
to return the output of the new impact those edits would have on the problem spa



face module 110,
functions, applying
e user edits the
djustment of the
ons are executed
ce.