MODULE *WorkflowRepair*

EXTENDS *WorkflowDefinition*

LOCAL INSTANCE *Utilities*

GIVEN the max workflow length of:

CONSTANT *MaxDepth*

```
e.g. MaxDepth == 5
```

such that it is a non-negative number

ASSUME $MaxDepth \in Nat$

with the following workflow distance definitions

$missingTaskTypes(W\_new,\ W\_old) \triangleq Cardinality(RAN(W\_old) \setminus RAN(W\_new))$

$additionalTaskTypes(W\_new,\ W\_old) \triangleq Cardinality(RAN(W\_new) \setminus RAN(W\_old))$

$missingTaskAmount(W\_new,\ W\_old) \triangleq BagCardinality(SeqToBag(W\_old) \ominus SeqToBag(W\_new))$

$additionalTaskAmount(W\_new,\ W\_old) \triangleq BagCardinality(SeqToBag(W\_new) \ominus SeqToBag(W\_old))$

$diffWorkflowLength(W\_new,\ W\_old) \triangleq Abs(Len(W\_new) - Len(W\_old))$

$diffTaskOrder(W\_new,\ W\_old) \triangleq$
LET
$task\_order\_diffs \triangleq [t \in (RAN(W\_old) \cap RAN(W\_new)) \mapsto$
$Abs(Sum(Indexes(W\_old,\ t)) - Sum(Indexes(W\_new,\ t)))$
$]$
$task\_order\_diffs\_bag \triangleq RestrictRangeWithPredicate(task\_order\_diffs,$ LAMBDA $n : n > 0)$
IN
$BagCardinality(task\_order\_diffs\_bag)$

and with the thus resulting set of all possible, valid workflows

$ValidWorkflows \triangleq \{$
$W \in BoundedSeq(TaskNames,\ MaxDepth) :$
LET $Validation \triangleq$ INSTANCE $WorkflowValidation$ WITH
$Tasks \leftarrow Tasks,$
$Connections \leftarrow Connections,$
$Workflow \leftarrow W$
IN $Validation!Errors = Validation!NoErrors$
$\}$

THEN the closest valid workflow is recommended as an alternative to the given workflow, in case the given workflow is invalid

$Recommendation \triangleq$
LET $Validation \triangleq$ INSTANCE $WorkflowValidation$ WITH

$Connections \leftarrow Connections,$
$Tasks \leftarrow Tasks,$
$Workflow \leftarrow Workflow$

IN

IF $Validation!Errors = Validation!NoErrors$
THEN $Workflow$
ELSE CHOOSE $W\_best \in ValidWorkflows$:
$\forall\, W\_worse \in ValidWorkflows : W\_best \neq W\_worse \Rightarrow$

1. minimize deletion of tasks while going from old to new wf

$\vee\ missingTaskTypes(W\_worse,\ Workflow)$
$\quad > missingTaskTypes(W\_best,\ Workflow)$
$\vee$
$\quad \wedge\ missingTaskTypes(W\_worse,\ Workflow)$
$\quad\quad = missingTaskTypes(W\_best,\ Workflow)$
$\quad \wedge$

2. minimize addition of tasks while going from old to new wf

$\quad\quad \vee\ additionalTaskTypes(W\_worse,\ Workflow)$
$\quad\quad\quad > additionalTaskTypes(W\_best,\ Workflow)$
$\quad\quad \vee$
$\quad\quad\quad \wedge\ additionalTaskTypes(W\_worse,\ Workflow)$
$\quad\quad\quad\quad = additionalTaskTypes(W\_best,\ Workflow)$
$\quad\quad\quad \wedge$

3. minimize reduction of task repetitions while going from old to new wf

$\quad\quad\quad\quad \vee\ missingTaskAmount(W\_worse,\ Workflow)$
$\quad\quad\quad\quad\quad > missingTaskAmount(W\_best,\ Workflow)$
$\quad\quad\quad\quad \vee$
$\quad\quad\quad\quad\quad \wedge\ missingTaskAmount(W\_worse,\ Workflow)$
$\quad\quad\quad\quad\quad\quad = missingTaskAmount(W\_best,\ Workflow)$
$\quad\quad\quad\quad\quad \wedge$

4. minimize increase of task repetitions while going from old to new wf

$\quad\quad\quad\quad\quad\quad \vee\ additionalTaskAmount(W\_worse,\ Workflow)$
$\quad\quad\quad\quad\quad\quad\quad > additionalTaskAmount(W\_best,\ Workflow)$
$\quad\quad\quad\quad\quad\quad \vee$
$\quad\quad\quad\quad\quad\quad\quad \wedge\ additionalTaskAmount(W\_worse,\ Workflow)$
$\quad\quad\quad\quad\quad\quad\quad\quad = additionalTaskAmount(W\_best,\ Workflow)$
$\quad\quad\quad\quad\quad\quad\quad \wedge$

5. minimize ordering difference of matching tasks in old and new wf

$\quad\quad\quad\quad\quad\quad\quad\quad \vee\ diffTaskOrder(W\_worse,\ Workflow)$
$\quad\quad\quad\quad\quad\quad\quad\quad\quad > diffTaskOrder(W\_best,\ Workflow)$
$\quad\quad\quad\quad\quad\quad\quad\quad \vee$
$\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge\ diffTaskOrder(W\_worse,\ Workflow)$
$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad = diffTaskOrder(W\_best,\ Workflow)$
$\quad\quad\quad\quad\quad\quad\quad\quad\quad \wedge$

6. minimize length difference between old and new wf

$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \vee\ diffWorkflowLength(W\_worse,\ Workflow)$

$$
\begin{aligned}
&> \mathit{diffWorkflowLength}(W\_best,\ Workflow) \\
\lor\ &\mathit{diffWorkflowLength}(W\_worse,\ Workflow) \\
&= \mathit{diffWorkflowLength}(W\_best,\ Workflow)
\end{aligned}
$$

```
e.g. Recommendation == << "EVI", "IVI" >>
```

such that it adheres to the expected structure

ASSUME $DOM(Recommendation) = 1 \,..\, Len(Recommendation)$ a proper tuple
ASSUME $\forall\, t \in RAN(Recommendation) : t \in$ STRING
ASSUME $\forall\, t \in RAN(Recommendation) : \exists\, task \in Tasks : task.name = t$