

Техническое задание для агента v0.1

Агента первой версии предполагается сделать настолько простым, что для его создания будет достаточно технологий, предложенных в рамках проекта и на данный момент полностью готовых к реализации.

Интерфейс агента

Агент существует в среде с дискретным временем. Каждый такт времени он получает из среды состояние своих рецепторов, затем вычисляет собственную реакцию и возвращает состояние своих эффекторов. Программное взаимодействие со средой осуществляется по средствам интерфейса:

```
// Универсальный интерфейс интеллекта агента v0.1
class IIntelligenceInterface01 {
public:
    typedef std::vector<int> TIoDataContainer;           // Контейнер
    данных эффекторов и рецепторов
    // Прожить 1 такт времени, [in] receptors - состояние
    рецепторов, [out] effectors - состояние эффекторов
    virtual void TimeStep(const TIoDataContainer& receptors,
    TIoDataContainer& effectors) = 0;
};
```

Параметры агента

У агента имеются фиксированные (неизменные в течении жизни) количества эффекторов и рецепторов, которые задаются средой при создании агента. Каждый рецептор и эффектор может находиться любом состоянии в диапазоне [0, MaxVal] где MaxVal - фиксированное в течении жизни число, индивидуальное для каждого рецептора и эффектора. Количество состояний задаются средой при создании агента. В качестве конкретных MaxVal не рекомендуется брать числа больше 10. **Рецептор с индексом 0 является оптимизируемым** поэтому агент будет действовать так, чтобы значения этого рецептора были максимальны.

Реализация агента

```
// Реализация интеллекта агента v0.1
class CIntelligenceV01 : public IIntelligenceInterface01 {
public:
    typedef std::vector<unsigned int> TIoLimitsContainer; //
    Контейнер данных о допустимых диапазонах IO данных
    // receptorsLimits - количества состояний для каждого рецептора,
    // effectorsLimits - количества состояний для каждого
    эффектора
    CIntelligenceV01(const TIoLimitsContainer& receptorsLimits,
    const TIoLimitsContainer& effectorsLimits);
    // IIntelligenceInterface01
};
```

```

virtual void TimeStep&#40; const TIoDataContainer& receptors,
TIoDataContainer& effectors &#41;;

private:
    const TIoLimitsContainer receptorsLimits;
    const TIoLimitsContainer effectorsLimits;

    typedef std::vector<double> TDistributionType;           //      Тип
для статистических распределений
    typedef std::vector<TDistributionType> TDistributionsContainer;
//      Контейнер из статистических распределений
//      Кластер статистической модели мира
    struct CCluster &#123;
        CCluster&#40; int receptorsCount, int effectorsCount &#41;;

        TDistributionsContainer ReceptorsStatistics;
        TDistributionsContainer EffectorsStatistics;
        TDistributionType ContextStatistics;
        TDistributionType::value_type Weight;
        TDistributionType::value_type Expectation;
    &#125;;

    typedef std::vector<CCluster> TClustersContainer;
    TClustersContainer worldModel;
    TDistributionType context;
    TIoDataContainer prevStepEffectors;

        //      Описание кандидата нового состояния эффектора
    struct CEffectorVariant &#123;
        CEffectorVariant&#40; TDistributionType::value_type
likelihood, int testedStateIndex,
        TIoDataContainer::value_type variant &#41;;

        const TDistributionType::value_type Likelihood;
        const int TestedStateIndex;
        const TIoDataContainer::value_type Variant;
    &#125;;

    //      Компаратор кандидатов
    struct CEffectorVariantComparer : public
std::binary_function<CEffectorVariant, CEffectorVariant, bool> &#123;
        bool operator&#40;&#41;&#40; const CEffectorVariant& left,
const CEffectorVariant& right &#41; const;
    &#125;;

    //      Массив приоритетных очередей
    typedef std::vector< std::priority_queue<CEffectorVariant,
std::vector<CEffectorVariant>,
        CEffectorVariantComparer> > TVariantQueuesContainer;

    //      Описание протестированного состояния эффектора

```

```
typedef std::vector< std::vector<bool> > TVariantsContainer;
class CEffectorsTestedState &#123;
public:
    CEffectorsTestedState&#40; const TIoDataContainer& effectors
&#41;;
    //          породить новую комбинацию
    CEffectorsTestedState&#40; CEffectorsTestedState& baseState,
        int effectorIndex, TIoDataContainer::value_type
newVariant &#41;;
    const TIoDataContainer& Effectors&#40;&#41; const &#123;
return effectors; &#125;
    const TVariantsContainer& TestedVariants&#40;&#41; const
&#123; return testedVariants; &#125;

private:
    TVariantsContainer testedVariants;          //
протестированные комбинации
    TIoDataContainer effectors;                //          состояние
эффекторов
    &#125;;
    typedef std::vector<CEffectorsTestedState> TTestedStatesContainer;

    void calculateNextStepContext&#40; const TIoDataContainer& receptors,
TDistributionType& adArchive &#41; const;
    void updateStatistics&#40; const TIoDataContainer& receptors, const
TDistributionType& adArchive &#41;;
    void findBestReaction&#40; const TDistributionType& adArchive,
TIoDataContainer& effectors &#41; const;
&#125;;
```

From:
<http://www.ailab.ru/wiki/> - AI Lab Wiki

Permanent link:
http://www.ailab.ru/wiki/doku.php?id=%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8:mumla:%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B4%D0%BB%D1%8F_%D0%B0%D0%B3%D0%B5%D0%BD%D1%82%D0%B0_v0.1

Last update: 2021/04/07 21:33